ARABIC SPEECH PROCESSING:
SYLLABIC SEGMENTATION AND SPEECH RECOGNITION

By

ABDULHADI SADOUN AL-OTAIBI
Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

October 1988

The University of Aston in Birmingham

ARABIC SPEECH PROCESSING: SYLLABIC SEGMENTATION AND SPEECH RECOGNITION

Abdulhadi Sadoun Al-Otaibi

Doctor of Philosophy 1988

SUMMARY

A detailed description of the Arabic Phonetic System is given. The syllabic behaviour of the Arabic language is highlighted. Basic statistical properties of the Arabic language (phoneme and syllabic frequency of repetition) are included.

A thorough review of the speech processing techniques, used in speech analysis, synthesis and recognition applications are presented.

The development of a PC-based speech processing system is described. The system has proven to be a useful tool in Arabic speech analysis and recognition applications.

A sample spectrographic study of two pairs of Arabic similar sounds was performed. It is shown that no clear acoustical property exist in distinguishing between the phonemes /θ/ and /f/ except the gradual rise of F1 during formant movements (transitions).

The development of an automatic Arabic syllabic segmentation algorithm is described. The performance of the algorithm is tested with monosyllabic and multisyllabic words. An overall accuracy of 92% was achieved. The main parameters affecting the accuracy of the segmentation algorithm are discussed.

The syllabic units generated from applying the Arabic syllabic segmentation algorithm are utilized in the implementation of three major speech applications, namely, automatic Arabic vowel recognition system, isolated word recognition system and an acoustic-phonetic model for Arabic. Each application is fully described and its performance results are indicated.

KEYWORDS

ARABIC PHONETIC SYSTEM, PC-BASED SPEECH PROCESSING SYSTEM, ARABIC SYLLABIC SEGMENTATION ALGORITHM, ARABIC SPEECH RECOGNITION APPLICATIONS

DEDICATION

This thesis is dedicated to my parents who have always encouraged the pursuit of knowledge.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisors, Dr. R. L. Brewster and Dr. Y. El-Imam for their advice, guidance and support through the period of this study.

Sincere gratitude and deepest thanks to Dr. R. C. F. Tucker for his valuable assistance right at the start of this research work.

Special thanks to my advisor, Professor J. E. Flood, Department Head, for awarding me this opportunity to pursue for higher academic degree.

Special thanks also to Mr. C. Pereira for his assistance and patience in typing of the manuscript.

Contents

TABLES

FIGURES

CHAPTER 1

Chapter 1

INTRODUCTION

1.1 Introduction

The ultimate goal of research in speech processing is to enhance our knowledge about speech sounds associated with languages and how speech is perceived. Such knowledge is essential if one attempts to build systems capable of man-machine communication. To acquire such knowledge one would also have to solve numerous and complex problems relating to thought development, vocal articulators, additive noise from media, and the speech perception process. Much of the basic research in speech processing has been carried out during the early fifties and late sixties. Successful systems designed for speech analysis and synthesis were demonstrated then. From the mid-seventies until today, speech processing areas have undergone unprecedented growth due to advancement in computer technologies, enabling fast and accurate computation to be achieved. Most speech research carried out today is either basic, i.e., validation of the basic findings due to better analytical tools, or applied research aimed at producing better systems for speech synthesis and recognition by carrying out thorough

investigations relating to the phonology of the language, and studies in speech perception.

1.2 Objectives

The primary goal of this investigation is to perform research in Arabic speech processing with emphasis on Arabic speech recognition. To fulfill the stated objective the following tasks were performed:

1. Literature review covering the following items:
   a. Arabic phonological system.
   b. Signal processing techniques applied in speech processing.
   c. Speech synthesis and recognition techniques.
   d. Arabic speech processing research efforts.
2. Development of a PC-based speech processing system suitable for performing studies in speech processing.
3. Spectrographic studies of the Arabic phonetic system such as determining formant and bandwidth values of Arabic vowel system.
4. Development of an Arabic syllabic segmentation algorithm.
5. Implementation of Arabic speech recognition applications using the developed Arabic syllabic segmentation algorithm. Applications include:

a. Automatic Arabic vowel recognition system.

b. Isolated Arabic word recognition system.

c. An acoustic-phonetic module for knowledge based speech recognition strategy.

An acoustic-phonetic approach is adopted based on syllabic units. The main motivation for adopting this strategy is the speaker independent property and capability for large recognition vocabulary.

The proposed strategy for the isolated Arabic word speech recognition system is shown in Figure 1. As seen from Figure 1, the system consists of several stages of which the most important is the segmentation stage. An algorithm was developed that acts on the acoustic signal segmenting it into discrete Arabic syllabic units. The algorithm uses the short time energy for the segmentation purposes.

1.3 *Thesis Structure*

This thesis examines the fundamentals of Arabic speech processing with emphasis on speech recognition. To facilitate research in speech processing, the development of a PC-based speech processing system is reported. The syllabic structure and behavior of the Arabic language is exploited in the development of an Arabic syllabic segmentation algorithm. This segmentation algorithm was utilized in three important aspects of Arabic

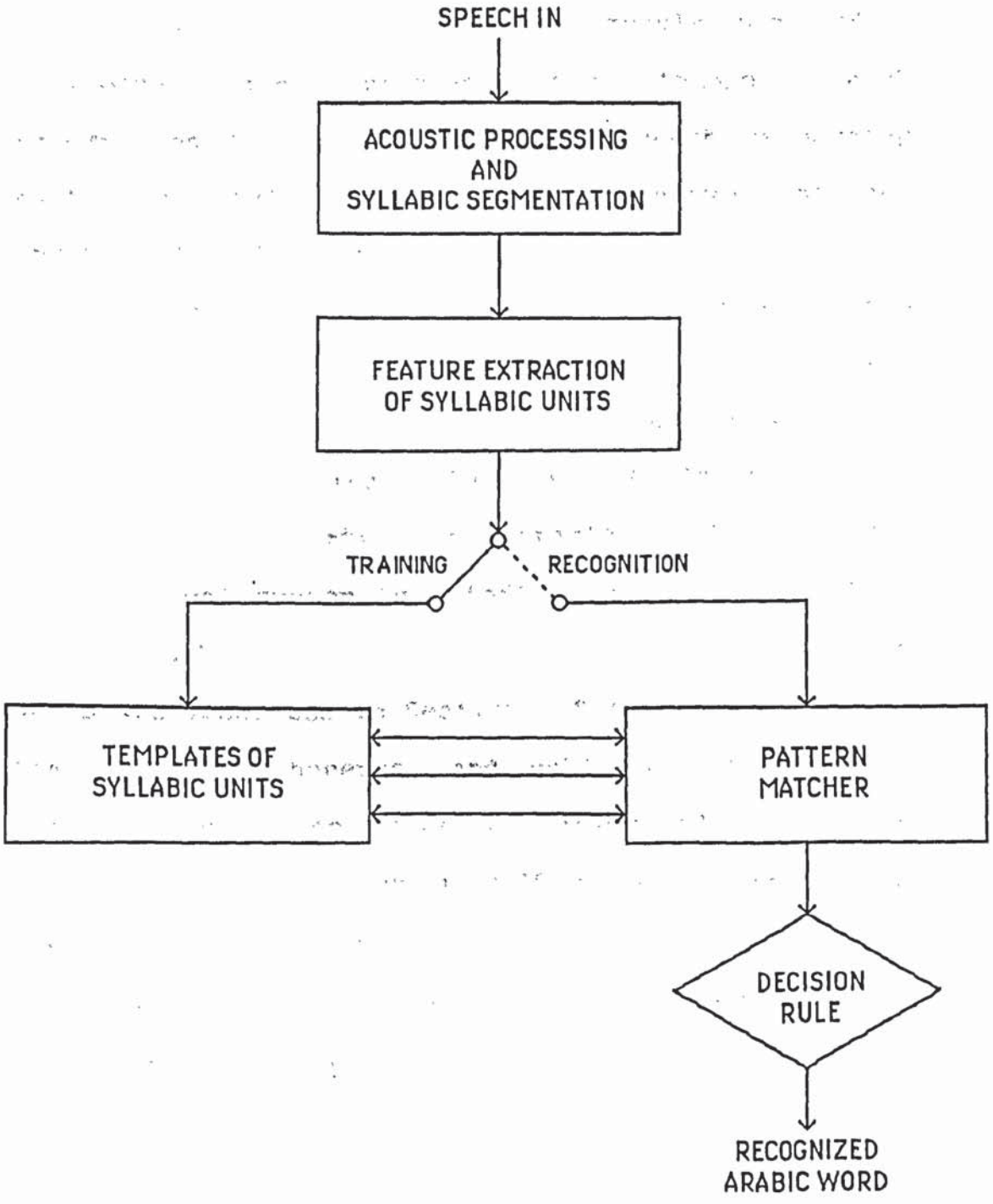


FIGURE 1

PROPOSED STRATEGY FOR THE ISOLATED ARABIC WORD RECOGNITION SYSTEM.

speech recognition applications namely, automatic vowel recognition system, isolated word recognition system and an acoustic-phonetic module part of a knowledge based strategy for Arabic speech recognition system. The PC-based speech processing system was used in implementing the above mentioned speech recognition applications.

Chapter 2 contains a detailed review of the Arabic phonetic system. In Sections 2.2, 2.3, 2.4 and 2.5 Arabic phonemes (consonants and vowels) are described and listed. Section 2.6 describes the Gemination property of Arabic. The Arabic morphological system is briefly described in Section 2.7. Arabic syllable types and structures within words are described in Section 2.8 together with some statistics on phonemes, and syllables repetitions. Arabic accentuation rules are described in Section 2.9. Finally, in Section 2.10 Arabic speech sound classifications are presented.

Chapter 3 contains a complete literature survey on speech analysis, synthesis and recognition techniques applied today. In Section 3.1, the speech production mechanism is described. Different types of speech synthesis methods are described in Section 3.2. Similarly, speech recognition strategies are presented in Section 3.3. A review of Arabic speech processing research efforts are presented in Section

3.4. Finally, different speech processing applications are reported in Section 3.5.

Chapter 4 reports the development of a PC-based speech processing system, which is vital for conducting research in speech processing (speech analysis, synthesis and recognition). The main hardware system architecture and functions are presented in Sections 4.2, 4.3, 4.4 and 4.5. Short time speech analysis techniques are reported in Section 4.6. General discussions on system uses are presented in Section 4.7.

Chapter 5 examines the acoustical properties of some Arabic similar sounds. Facilities used to generate speech spectrograms are described in Section 5.2. Arabic phonemes used and their measured acoustical parameters are reported in Section 5.3, 5.4 and 5.5. Conclusion is given in Section 5.6.

Chapter 6 describes the development of the automatic Arabic syllabic segmentation algorithm. In Section 6.1, the concept of speech segmentation is presented. The suitability of Arabic syllables as units for speech recognition is reported in Section 6.2. Syllabic segmentation for the English language is described in Section 6.3. Characteristics of the Arabic language, Arabic syllabic definition and the development of the Arabic syllabic segmentation algorithm are described in Sections 6.4.1 - 6.4.3.

Syllabic segmentation results are described in Sections 6.7 and 6.8. Final conclusion is report in Section 6.9.

Chapter 7 reports and describes three important aspects of Arabic speech recognition applications. All three aspects make use of the developed Arabic syllabic segmentation algorithm. In Section 7.2 - 7.2.6 the automatic Arabic vowel recognition system is described with vowel recognition results reported. The proposed strategy for isolated Arabic word recognition system is presented in Section 7.9.2. System description and recognition results are given in Sections 7.9.3 - 7.9.6. In Section 7.9.8, an acoustic-phonetic module to be part of a knowledge based strategy for Arabic speech recognition is reported.

Chapter 8 is an overall conclusion of the entire research work attempted and suggestions for further work are outlined.

CHAPTER 2

Chapter 2

ARABIC PHONETIC SYSTEM

2.1 Introduction

Arabic, in similarity with other languages, is characterized by a discrete set of 38 phonemes. Phonemes are generally considered to be the smallest unit of speech sound. Since many realizations of a specific phoneme exist due to variations of context but carrying the same meaning, a phoneme may be best defined to be the common dominator of all realizations of specific sound. The phonemes of any language are established by means of a minimal pair. Speech sound units (phonemes) are classified into either consonants or vowels. The classification criteria is based on articulatory, acoustic and contextual information.

2.2 Arabic Alphabet

The Arabic alphabet consists of 28 letters shown in Table 1. The letter alif has no sound of its own but is used merely to support hamzah /ʔ/, to lengthen a preceding vowel, and at the end of third person plural of verbs. Arabic writing is performed from right to left, where letters take different forms depending on their position in words (contextual analysis). Some letters are similar to others except

Arabic Letters	Pronounciation	English Equivalent	IPA
ء	hamzah	-	ʔ
ب	baa	b	b
ت	* taa	t	t
ث	* thaa	th	θ
ج	jeem	j	ʒ
ح	haa	h	ħ
خ	khaa	kh	x
د	* daal	d	d
ذ	* dhaal	dh	ð
ر	* raa	r	r
ز	* zaa	z	z
س	* seen	s	s
ش	* sheen	sh	ʃ
ص	* ṣaad	ṣ	s
ض	ḍaad	ḍ	d
ط	* ṭaa	ṭ	t
ظ	* ẓaa	ẓ	ð
ع	* ayn	-	ʕ
غ	ghayn	gh	ɣ
ف	faa	f	f
ق	qaaf	q	q
ك	kaaf	k	k
ل	* laam	l	l
م	meem	m	m
ن	* noon	n	n
ه	haa	h	h
و	waw	w	w
ي	yaa	y	j

* indicates sun letters.

Table 1

The Arabic Alphabets.

for a dot (diacritical point) placed above or beneath them, e.g., daal /d/ and dhaal /ð/. Arab linguists characterize the Arabic letters into two classes, namely, sun letters and moon letters [1]. Sun letters are indicated with an asterisk in Table 1. Sun letters are those which when preceded by the prefix alif laam /ʔl/ in nouns, the laam /l/ consonant is not pronounced.

2.3 *Arabic Consonants*

There are 28 Arabic consonants shown in Table 1. Most consonants have distinct pronunciation rules. Alif is considered a consonant when it carries a long vowel. The consonant daad /ḍ/ is peculiar to the Arabic language because non-Arabs cannot pronounce it correctly. The following consonants are pronounced like English:

baa	/b/	sheen	/ʃ/
taa	/t/	faa	/f/
thaa	/θ/	kaaf	/k/
jeem	/ʒ/	laam	/l/
daal	/d/	meen	/m/
dhaal	/ð/	noon	/n/
raa	/r/	haa	/h/
zaa	/z/	waw	/w/
seen	/s/	yaa	/j/

The following consonants have no English equivalents and are considered as foreign to English:

hamzah	/ʔ/	taa	/t/
haa	/h/	zaa	/ð/
khaa	/x/	ayn	/ʕ/
saad	/S/	ghayn	/ɣ/
daad	/d/	qaaf	/q/

2.4 Arabic Vowels

Vowels are not normally indicated in written books or magazines, unless the correct pronunciation of the word is to be ensured. Every consonant in vocalized Arabic text is provided with a vowel sign. These vowel signs are called (حركات) "Harakat" in Arabic meaning 'movement'. Thus a consonant with a vowel sign is called (متحرك) "Mutaharrik" in Arabic meaning 'moving', and a consonant without a vowel sign is called (ساكن) "Sakin" in Arabic meaning 'still'.

Arabic has three short and three long vowels [2]. The short vowels are:

* Fathah /a/, a small diagonal stroke is placed over consonants (ـَ).
* Dammah /u/, a small diagonal comma is placed over consonants (ـُ).
* Kasrah /i/, a small diagonal stroke placed under consonants (ـِ).

The long vowels are /a:/, /u:/ and /i:/ which result from short vowels being followed by the consonants alif, waw /w/ and yaa /j/, respectively.

2.5 Arabic Dipthongs

When two vowel-like phonemes occur one after the other in the same syllable, a vowel-like sound occurs called a dipthong.

Arabic has two types of dipthongs /aj/ and /aw/ which result when the short vowel fatha /a/ is followed by the consonants waw /w/ and yaa /j/ as in the words /xawf/ "afraid" and /bajt/ "house".

2.6 Gemination

A feature of the Arabic language is that every consonant can be geminated, i.e., pronounced twice. The sign 'ـّـ' is placed over the consonant to indicate gemination. Geminated consonants are pronounced at least twice as long as its single counterpart when they occur in the middle parts of speech and are characterized by greater muscular tension in the articulatory organs. Their duration is shorter when they occur in final word positions.

2.7 Arabic Morphological System

The Arabic morphological system is well understood and has been the subject of research for a good number of years by Arab scholars. Any Arabic word, e.g., nouns, verbs, or adjectives can essentially be derived from three types of roots,

namely, triradical, quadradical and five radical. Different patterns of these roots constitute different Arabic words, but not all Arabic words can be derived from these patterns, irregularity exist. Each pattern usually has a semantic (meaning) and syntactic (grammatical) value. Patterns are formed by one or more of the following changes added or deleted to the basic pattern:

* Prefixing.
* Suffixing.
* Changing patterns of vowels.
* Adding medial consonants.

Arabic verbs are derived from triradical and quadradical roots. Triradical roots are the most frequently used in Arabic and they constitute more then 63% of Arabic roots [2]. Arabic nouns and adjectives are also defined from similar patterns.

Having a lexical dictionary carrying information on morphology and syntax is an essential stage in speech synthesis and recognition systems.

2.8 Arabic Syllables

A syllable is a unit of speech which is larger than the basic units of speech (phonemes), and consist of short or long vowels associated with one or more consonants. The Arabic language has six

different types of syllables, namely, CV, CVV, CVC, CVVC, CVCC and CVVCC. Those syllables ending with a vowel are termed as open syllables, and those ending with a consonant are termed as closed syllables.

The most common syllables in Arabic are CV, CVV and CVC. CVCC and CVVCC are not common and they occur in isolation or in speech final pausal forms. Arabic words may contain from one syllable as in /ba/ to seven syllables. Syllables are also divided into short, medium and long. Short syllables have one consonant plus a short vowel as in /da/. Medium syllables consist of a long vowel plus a consonant as in /ja:/, or a short vowel plus two consonants as in /jad/. Long syllables consist of a long vowel plus two consonants as in /ba:b/, or of a short vowel plus three consonants as in /darb/.

Several statistical studies have been made for the determination of entropy and redundancy of the Arabic language [1, 3], however no study has been made to determine and analyze the probabilities of occurrence of Arabic word syllabic structures forms. Hence, to get an idea of the most probable Arabic words syllabic structure forms, a sample Arabic text consisting of 510 words (Appendix A) was entered into the personal computer (PC). The sample text was taken from an article that appeared in one of the Kuwaiti daily newspapers.

A special provision was made in the PC environment to allow for the Arabic vowel signs (diacritic marks) to be entered along with Arabic text. The text was analyzed to determine the following:

1. Frequency of occurrence of Arabic consonants and vowels.
2. Frequency of occurrence of Arabic syllables.
3. Arabic word syllabic structures.

Tables 2 and 3 show the frequency of occurrence of Arabic consonants and vowels, these results are similar to those reported in the literature [3, 4]. Table 4 shows the frequency of occurrence of Arabic syllable types. It is seen from Table 4 that the most frequent syllable types are the CV, CVC and CVV, accounting for almost 90% of the time. Table 5 shows the ten most reccurrent word syllabic type structures in ascending order.

2.9 Arabic Stress Rules

There are two types of stress, one is called the primary stress which is inherent and acts at the word level, while the other type is global, i.e., it acts at the phrase or the sentence level. The primary stress depends on the syllabic structure of the word.

When an Arabic word has more than one syllable, then one of these syllables will receive the primary stress, which is defined to be the sudden activity

Arabic Consonants	IPA	Frequency of Repetition
	pause	0.17846
ء	ʔ	0.17501
ب	b	0.02551
ت	t	0.03696
ث	θ	0.00475
ج	ʒ	0.01136
ح	ħ	0.01555
خ	x	0.00596
د	d	0.02914
ذ	ð	0.00465
ر	r	0.04152
ز	z	0.00484
س	s	0.02327
ش	ʃ	0.00754
ص	ṣ	0.00791
ض	ḍ	0.00698
ط	ṭ	0.00121
ظ	ð̣	0.00680
ع	ʕ	0.02569
غ	ɣ	0.00410
ف	f	0.01918
ق	q	0.02206
ك	k	0.02147
ل	l	0.09551
م	m	0.04664
ن	n	0.04180
ه	h	0.05083
و	w	0.04068
ي	j	0.05362

Table 2

Frequency of occurence of consonants.

Arabic Vowels	IPA	Frequency of Repetition
فتحة	a	34%
فتحة طويلة	a:	15%
كسرة	i	24%
كسرة طويلة	i:	5%
ضمة	u	11%
ضمة طويلة	u:	3%
Arabic Dipthongs	**IPA**	**Frequency of Repetition**
فتحة ـ ى	aj	3%
فتحة ـ و	aw	5%

Table 3

Frequency of occurrence of vowels and dipthongs.

Syllable Type	Frequency of Repetition
CV	50%
CVC	22%
CVV	16%
CVVC	6%
CVCC	6%

Table 4

Frequency of occurrence of syllable types.

Syllabic Structures
CV
CVC
CVV
CVC CV CVV CV
CVC CV
CVC CV CV
CV CVV CV
CV CV CV
CVC CVV CV
CV CV

Table 5

Syllabic Word Structures.

that the vocal organism undergoes during vocalization of a particular syllable, which in turn will have an effect on syllable shape, duration and intensity. The rules for applying accentuation in Arabic are as follows [4]:

1. If the word is made of only one syllable, then naturally stress is placed on that syllable only.
2. If the word is made of two syllables, then stress is placed on the second syllable.
3. If the word is made of three syllables and the second syllable is that of CVV or CVC or CVVC or CVCC, then stress is placed on the second syllable.
4. If the word is made of three syllables and the second syllable is that of CV (short), then stress is placed on the third syllable.
5. Stress is never placed beyond the third syllable.
6. /ʔl/ is not considered as a syllable in this definition.

2.10 Arabic Speech Sound Classification

Arabic speech sound can generally be classified according to any of the following categories:

2.10.1 Places and Manners of Articulation. Speech sounds are produced as a result of the air stream passing through the vocal tract facing some kind of obstruction. Hence, consonants are

classified according to the places and manners of these obstructions. A book by Peter Ladefoged [5] gives a detailed account of this. Table 6 shows the phonetic chart of the Arabic consonants, with the appropriate speech sound classifications indicated.

2.10.2 Voiced/Unvoiced Classification. Voiced speech sounds are produced when the vocal cords are set into vibration as a result of the air stream passing through them. Unvoiced speech sounds are produced as a result of the air passing through the vocal tract, facing at some point along the vocal tract some kind of obstruction, hence causing the escaping air to become turbulent and produce hissy sounds like sha /ʃ/. Voiced/unvoiced Arabic consonant classification is also shown in Table 6.

2.10.3 Emphatic/Non-Emphatic Classification. The group of consonants that are articulated with the tongue being as broad as possible and rigid at its root towards the hard palate are called "emphatic consonants", because the sounds generated by them tend to be very well emphasized and intensive. In Arabic, there are three types of emphatic consonants [6], they are:

1. Totally emphatic, i.e., /S/, /d/, /ʧ/, /ʁ/, and /ˈ/. These group of consonants are well distinguished from their equivalent non-emphatic counter parts, i.e., /s/, /d/,

	Phoneme	IPA	Nasal	Stop-Plosive	Fricative	Glide	Semi-Vowel	Emphatic	Non-Emphatic	Voiced	Unvoiced
Bilabial	ب م و	b m w	x	x			x		x x x	x x x	
Labio Dental	ف	f			x				x		x
Inter Dental	ث ذ ظ	θ ð ðˤ			x x x			x	x	x x	x x
Alveolar Dental	ت د ز س ش ص ض	t d z s ʃ sˤ dˤ		x x x x	x x x			x x x	x x x x	x x x x	x x x
Alveolar	ر ل ن	r l n	x			x x x		x x	x x	x x x x	
Palatal	ج ي	ʒ j ʎ			x x		x		x x	x x	x
Velar	خ غ ك	x ɣ k		x	x x			x x		x	x x
Pharyngeal	ح ع	ħ ʕ			x x					x	x
Glottal	ه ء	h ʔ		x	x						x x
Uvular	ق	q		x				x			

Table 6
Arabic Phonetic Chart.

/t/, /θ/.

2. Partially emphatic, i.e., /x/, /ɣ/, and /q/. These groups of consonants have no equivalent counter-parts.

3. Temporary emphatic. Only the consonant /r/ can sometimes be emphatic depending on context and will be produced with an emphatic-like sound.

CHAPTER 3

Chapter 3

REVIEW OF CURRENT RESEARCH

3.1 *Speech Production*

The speech production apparatus consists of three systems [7], namely, the respiratory system which provides the source of energy, the laryngeal system which is responsible for the transduction of the moving air into acoustic energy (phonation), and the supraglottal system which incorporates the oral and nasal cavities, the tongue, lips and jaw. Figure 2, shows the main organs of the speech production apparatus; the vocal organs include the pharynx, nose and mouth. The shape of the vocal tract can be varied extensively by moving the tongue, the lips and other parts of the tract (articulators). Every shape taken by the vocal tract will imply a different acoustic property. The steady stream of air pulses will cause the vocal cords to resonate (vibrate) in different ways depending on the positions assumed by the different articulators. Hence, various varieties of sounds are generated. The following are the major classifications of sounds generated:

1. *Vowels*

   A vowel sound is generated when the air passage is open. Different mouth and tongue

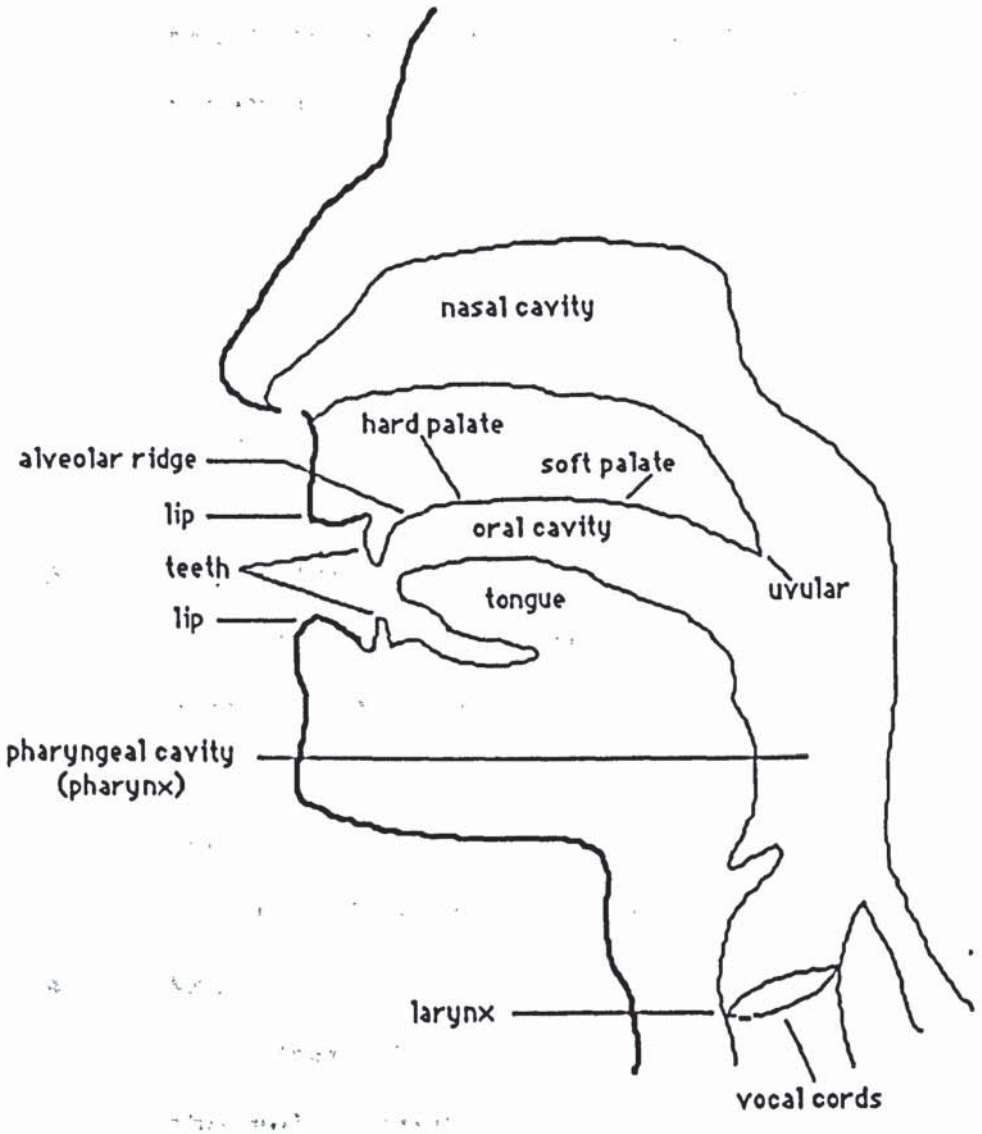


FIGURE 2

THE ORGANS OF SPEECH PRODUCTION.

shapes produce different vowel sounds such as /a/, /i/ and /u/.

2. ***Plosives***

The passage of air flow through the vocal tract is blocked by a complete closure usually towards the front of the vocal tract. Pressure builds up behind the closure and when the air is suddenly released a plosive sound is produced. In Arabic, five types of closure occur resulting in five sets of plosives, e.g., closure made by the lips, producing a bilabial plosive /b/.

3. ***Fricatives***

Partial closure (constriction) at some point in the vocal tract causes the air stream to escape through a narrow channel, hence, the escaping air becomes turbulent, producing hissy sounds called fricatives, e.g., /s/.

4. ***Nasal***

Complete closure of the mouth by lowering the velum causes the air to escape through the nose, producing nasal sounds, e.g., /m/.

Table 6 shows the complete classification of Arabic sounds.

3.2 *Speech Synthesis*

Speech sounds are produced as a result of the steady stream of air expelled from the lungs during exhalation cycles through the vocal tract facing some kind of obstruction. Different sounds are produced depending on the type of obstructions as mentioned before. Speech synthesis is defined as "the process of speech sound production from a phonetic transcription of messages" [8]. The speech synthesizer's job is to construct the speech signal from a set of parameters which are extracted during speech-analysis sessions. A speech synthesizer must reproduce not only the characteristics of sounds which most closely represent the steady state parametric characteristics of each phoneme, but also the dynamics of vocal tract motion as it progresses from one phoneme to another [9]. This fact highlights the difference between synthesis methods found today. The size of the "building blocks" determines the concatenation rules complexity, i.e., the larger the units (dictionary) the less complex the concatenation rules. A compromise has to be chosen between the size of the "building blocks" units and the number of rules governing them [8].

There are two speech synthesis techniques, namely, speech synthesis from pre-recorded speech elements (waveform coding), and speech synthesis by

reconstruction using parameters of a source-filter model, which can be realized in the frequency or time domain. Each technique is favoured depending on the application; hence for a text-to-speech system where an unlimited vocabulary is required, the source-filter model speech synthesizer is best suited. Most synthesizers use source characteristic parameters such as pitch (fundamental frequency), intensity, degree of voicing, and model characteristic parameters such as formants, LPC (Linear Prediction Coding) coefficients and energy levels. A comparison of the different approaches and synthesis techniques are shown in Table 7.

3.2.1 *Speech Synthesis from Pre-recorded Words and Phrases.*

This is a very simple method of synthesis; speech is coded as whole words or phrases, therefore speech output is natural and highly intelligible. Memory consumed is very large due to the high bit rate of speech coding, i.e., 64 k bits/sec. assuming 8 kHz sampling frequency and 8 bit encoding of each speech sample. Several techniques are applied for speech data compression such as ADPCM (Adaptive Pulse Code Modulation) which can reduce the rate to 16 k bits/sec. [10]. ADPCM encodes the relative amplitude changes of speech samples rather than the absolute amplitude values, hence the storage requirement is very much reduced.

	Direct Waveform Coding	Source/Filter Model			
		Channel Vocoder	LPC	Formant	Articu-latory
Hardware Complexity	Simple	Average	Simple	Average	Complex
Itelligi-bility	Very High	High	High	High	Very High
Applica-cation	Limited	Limited	Wide	Wide	Rare
Memory Consump-tion	Very High	Average	Small	Small	Small
Trans-mission Rate	64 k bits/sec.	2400-9600 bits/sec.	1000-2000 bits/sec.	1000 k bits/sec.	50 k bits/sec.
Vocabulary	Limited	Limited	Limited	Unlimited	Unlimited
Problems	No control over parameters	U/UV excitation	U/UV excitation	Formant Tracking	Control
Publicity	Not Popular	Popular	Very Popular	Very Popular	Not Popular

Table 7

Comparison between different approaches and synthesis techniques.

3.2.2 *Speech Synthesis by Source Filter Model.*

The speech production apparatus can be modelled by a time-varying model consisting of a filter excited by a source of either a noise generator for voiceless sounds or a periodic generator for voiced sounds. This model is shown in Figure 3. Reconstruction of speech can be made in the frequency domain by techniques such as channel vocoder, linear prediction synthesis and formant synthesis, or in the time domain by articulation-model synthesis [11]. Examples of frequency domain synthesis techniques are:

1. *Channel Vocoder*

   The principle behind this technique is to find the power $|sf|^2$ seen by an appropriate temporal window (10 msec. in duration) for different frequency bands. The number of frequency bands vary between 12 and 14. In each frequency band the energy is extracted; hence 12-14 parameters are extracted during each frame. Additional parameters are the decision for voiced or unvoiced temporal window segments as well as the value of pitch for voiced temporal-window segments. Figure 4 shows the source-filter model of the channel vocoder.

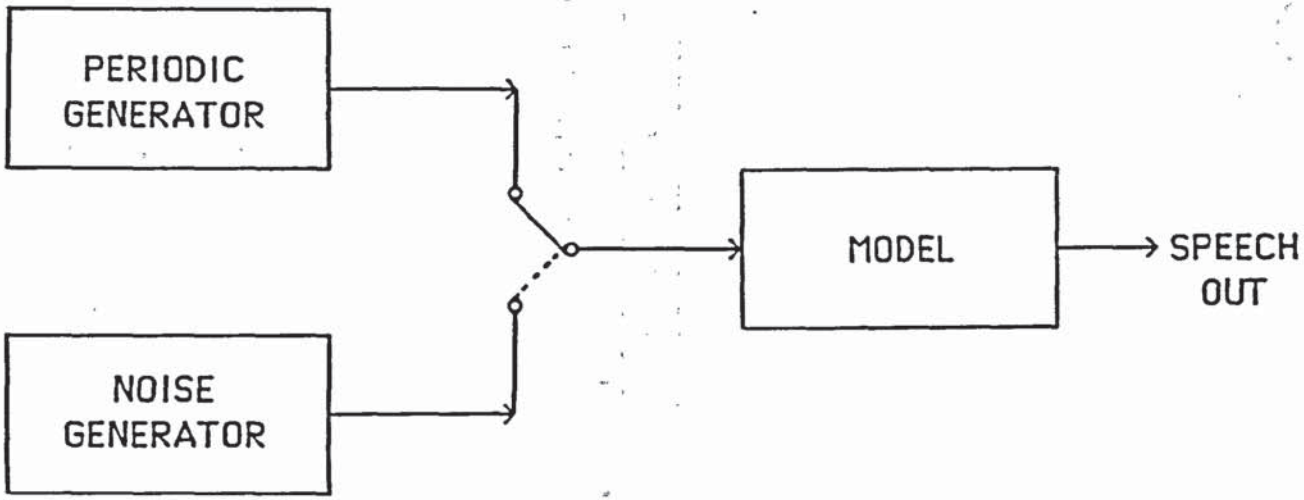


FIGURE 3

SPEECH SOURCE FILTER MODEL.

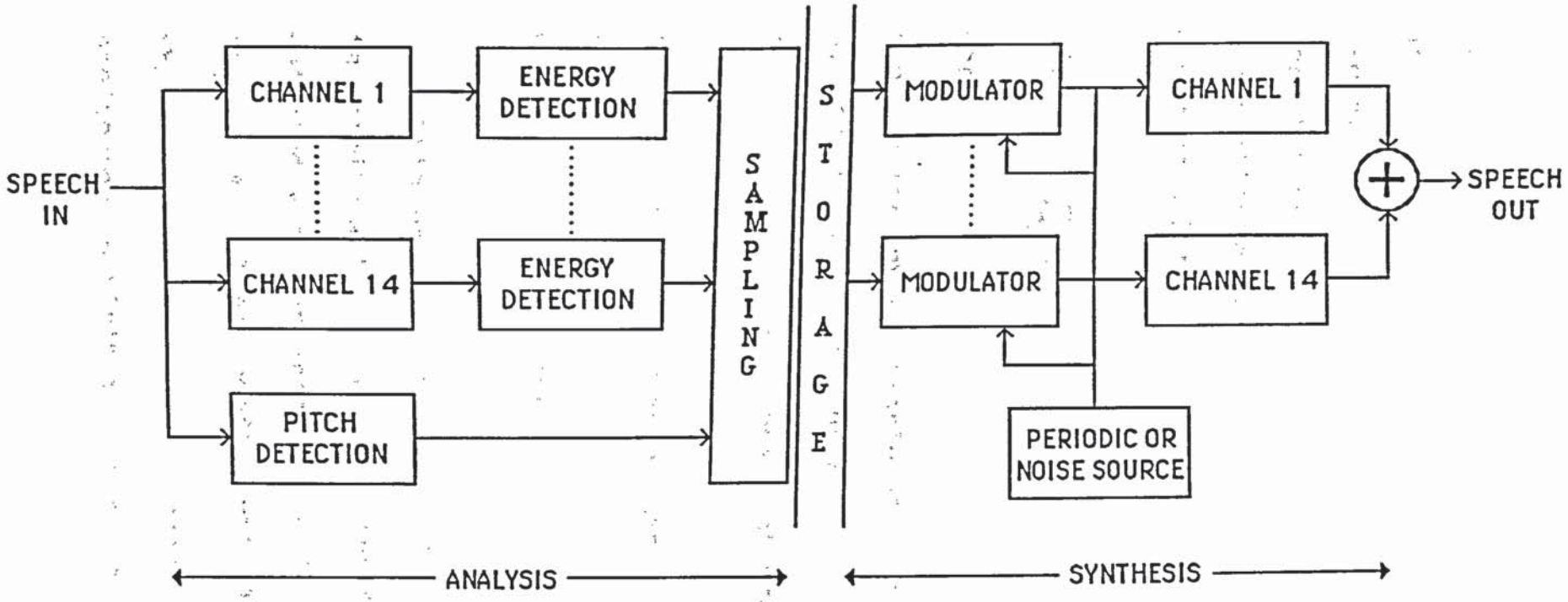


FIGURE 4

CHANNEL VOCODER.

2. *Linear Prediction Synthesis*

The literature is very rich in applications of Linear Predictive Coding (LPC) applied to speech processing [12]. This technique is very popular due to its computational simplicity. It draws on the high correlation between adjacent speech samples in assuming a linear relationship of the form:

$$s(n) = \sum_{i=1}^{P} a_i s(n-i) + GU(n) \quad \ldots \quad 1 \leq n \leq N \qquad (1)$$

where,

s(n) is the speech sample at time n,

a_i are the LP coefficients,

P is the number of LP coefficients,

G represent the gain,

N equals number of speech samples in one frame, and

U(n) is excitation function.

This equation states that a speech sample is predictable from the previous p samples weighted with the appropriate weighting factors ($a_1 \ldots a_p$). These coefficients are chosen such that the mean square error (original - predicted) is minimum. In minimizing the error

a set of p linear equations with p unknowns, $a_1 \ldots a_p$ are formed. Two methods are normally used for the solution of these equations, namely, the auto-correlation method and the covariance method [13]. Figure 5 shows a block diagram of the LPC synthesizer. In each time frame (10-25 msec.) the following parameters are extracted during analysis sessions:

* P number of Linear Prediction Coefficients (LPC).
* Amplitude (Gain).
* Voice/Unvoiced Flag.
* Pitch Period.

The size of P varies between 10-14. The advantage of LPC is that it separates the excitation source properties, i.e., pitch and amplitude from the vocal tract filter model which governs phonemic articulation, hence separating much of the prosodic information.

3. *Formant Synthesizer*

The formant synthesizer simulates the transfer function of the vocal tract H(z). The transfer function is characterized by the resonant frequencies, which are characteristics of phonemes. Hence, formant synthesizers are closely related to speech production as sound

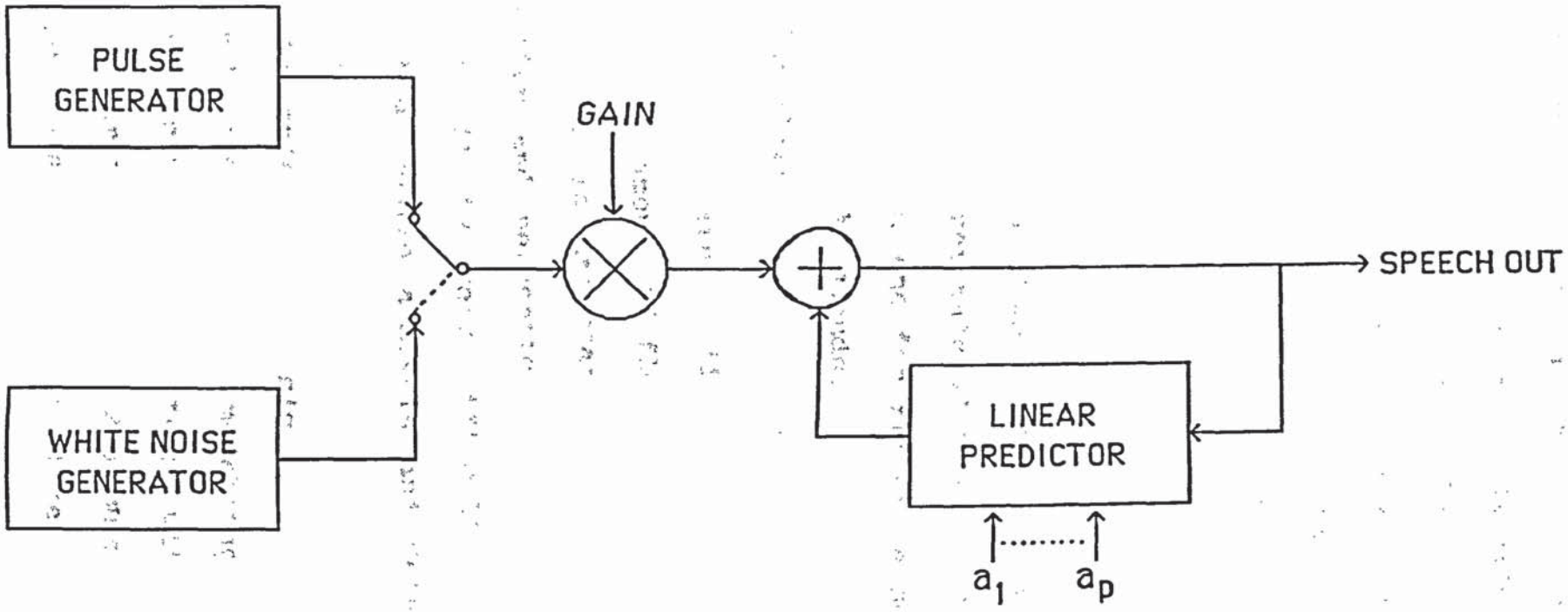


FIGURE 5

LPC SYNTHESIZER.

propagates from the glottis to the lips. Formant synthesizers can be implemented in cascade or parallel forms. Cascade formant synthesizers are not suitable for nasals because they assume an all-pole model where poles represent formants. A parallel cascade branch is added to cascade format synthesizers in order to synthesize nasality. Parallel formant synthesizers are more versatile because they allow more control over parameters. Figure 6 shows a block diagram of a parallel formant synthesizer made of three branches, one for nasal sounds, one for fricative sounds and one for vowel-like sounds.

The advantages of the formant synthesizer over others are: first, it allows a considerable amount of flexibility, and second, it provides an important insight of speech production and perception. The disadvantages are that it is very difficult to have a reliable automatic analysis and detection of formants.

3.2.3 *Articulatory Synthesis.*

This technique relies on modelling the vocal tract by a lossy acoustic tube with variable cross-sectional areas A (x, t). Propagation of sound waves through this tube is simulated by solving

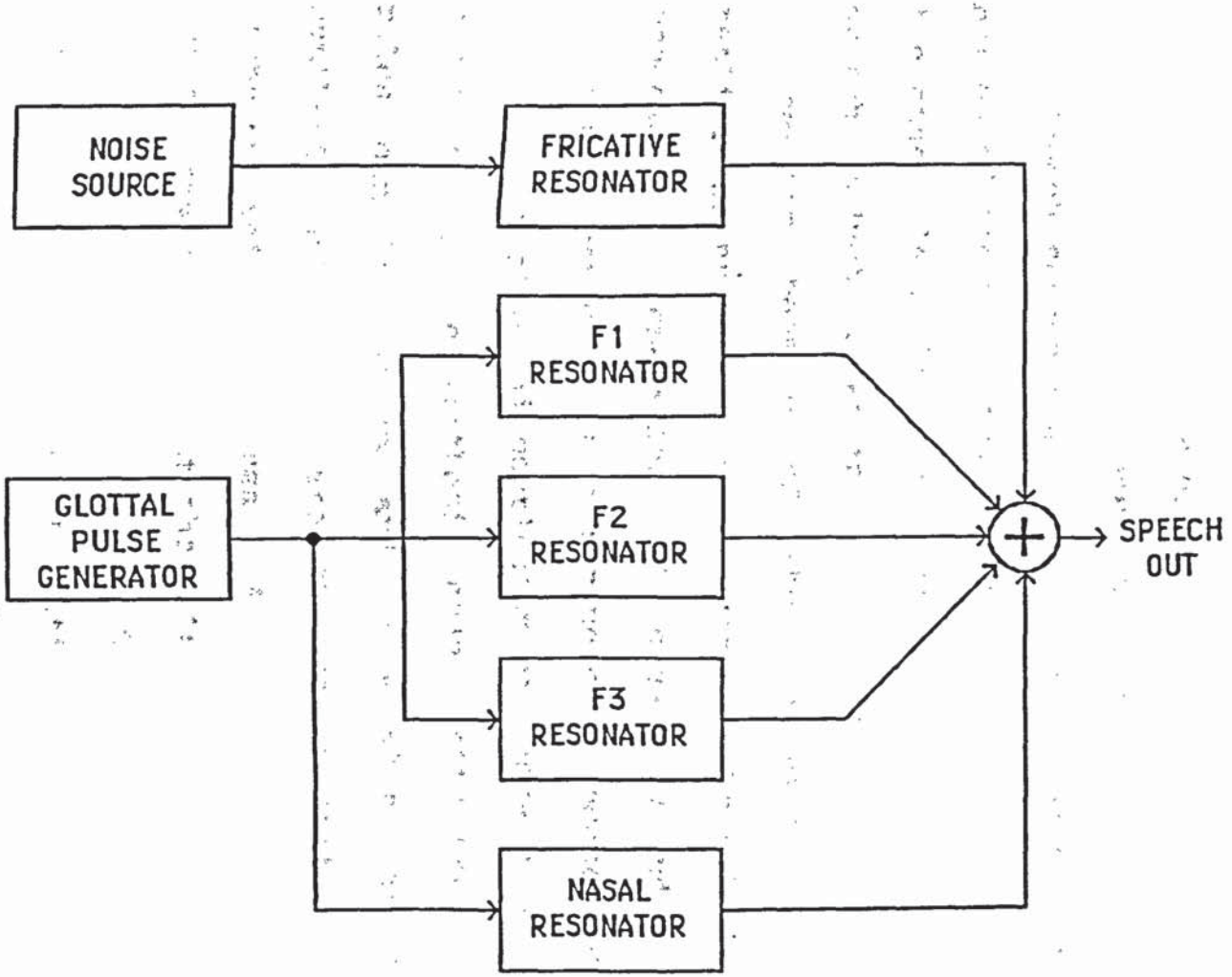


FIGURE 6

FORMANT SYNTHESIZER.

partial differential equations describing the physical phenomena [11]. The advantage of articulatory synthesis is the high-quality speech output, especially for vowels. The disadvantage of this technique is the complexities required to generate control parameters.

3.2.4 *Speech Synthesis by Rule.*

Speech sounds are produced by a near-continuous motion of the vocal tract apparatus from one side to the other [14]. Faithful computer simulation of the speech production mechanism is a very difficult task to perform because of the complexity in the realization process. Speech synthesis by rule is the process of generation of speech sounds from pre-stored sound units or parameters concatenated by rules. The size and complexity of those rules vary according to the synthesis sound units used. The rules can be complex and large when the sound synthesis units used are the basic units, i.e., phonemes and vice versa. A look at any speech spectrogram reveals that there is hardly any break in the speech patterns; hence intelligence is not carried in the phonemic information alone [15].

If we attempt to string phonemes together, then the resulting speech sound is very rough, not intelligible, and bumpy. It is well recognized that intelligibility for many consonants is associated

with the direction, extent and duration of formant movements. Liberman [16] pointed out that all transitions for a given consonant have a common feature, i.e., they start from one place called the characteristic locus. Knowing the first, second and third formant loci for all consonants is the key to unlock the syllables and make it feasible to write rules at the phoneme level. Liberman developed a minimum set of rules for the consonant and vowels.

At the Naval Research Laboratory [17], a set of letter-to-sound rules to guess the pronunciation of any word were developed using 50,000 commonly-used English words. The number of rules were 329. They were implemented on a program targeted for the Votan VS-6 phonetic synthesizer.

Similarly, the same strategies of letter-to-sound rules were followed which led to the development of a rule-based interactive tool for speech synthesis called "SRS" (Speech Research System) at Cornell University [18]. The idea behind these systems is the development of a practical set of text-to-phoneme rules.

Concatenation of viable sound units (words, syllables and phonemes) is the central problem for any synthesis-by-rule system. As stated earlier, the number of rules and their complexities is inversely proportional to the size of the dictionary. Hence,

numerous and complex rules are encountered in phonemic speech-synthesizer systems because we need rules that realize the transitions from one phoneme to the other. To reduce the size and complexity of the rules, sound units that are larger than the phonetic units can be used, where coarticulation effects (transitions) are intrinsic to them and need not be generated by rule. Holmes in 1964 developed the first phonetic synthesis-by-rule program [19]. The system is composed of a phonemic translator and a synthesizer. The program of the phonemic translator acts on text input and calculates for each 10 msec., 9 parameters (fundamental frequency, 3 formants, 3 amplitudes, state of switch for voicing and high-frequency formant). Parameter values for each phoneme are stored in a table which contains element duration, steady-state values and specifications of in-and-out transitions. Each phoneme has its own processing rank, and linear interpolation is carried out from the high-ranking phoneme to the lower ranking phoneme. Rank values are assigned in the order from 1-31, high rank numbers are given to phonemes characterized by transitions, hence stop consonants have the highest rank numbers, while vowels have lowest rank numbers.

Speech-synthesis systems based on interphoneme transition units (diphones) have been reported for English [20], and Arabic [21]. Other

speech-synthesis units such as VCV have also been reported with success, transitions from consonants to vowels and vice-versa is intrinsic to itself.

Because the number of vowels is small in Japanese, i.e., 5 vowels, the size of the dictionary of VCV units is 768. It is therefore feasible to realize for Japanese but not for English which has a dictionary size of 9072 (18 vowels, 28 consonants). For Arabic, the size of the dictionary would be 1044, which is attractive.

Speech synthesis by rule on syllabic units is not attractive due to the large number of syllabic units, i.e., Arabic would require theoretically about 78387 syllabic units. Because the influence of coarticulation effects diminishes when a syllable boundary is crossed [22], it is possible to split the syllable into two demisyllables. Due to language constraints, the number of demisyllables is relatively small; hence they are more favorable than syllables. An unlimited text-to-speech system is reported by Dettweiler [22] for the German language that has an inventory of 1650 demisyllables.

For Arabic, the number of the demisyllables would be 5220. This is still large, and needs to be reduced by either the vowel-substitution method, or by further splitting of the consonant clusters [22].

3.3 *Speech Recognition*

Speech recognition is an important topic because success in this area implies better man-machine interfaces which have the advantages of high speed, simple input to machines, and no requirements for prior training. To recognize fluent conversational speech with a vocabulary of thousands of words is not yet possible because of the following problems [23]:

1. ***Continuity Problem***

   Speech signals flow continuously without any pause or break in them, so it is very difficult to find any acoustical marks that indicate word or syllable boundaries.

2. ***Variability Problem***

   This is the biggest and most important problem, because it is affected by several factors such as the speaker's sex, age and accent. Sources of variability are the variation of times required to vocalize words and the continuity of speech. Hence, recognition algorithms must deal with pattern similarity rather than absolute identities.

3. ***Ambiguity Problem***

   There is no acoustic difference between the words "to" and "too", or between "grey tape" and "great ape". Hence, recognition

algorithms must include a component which resolves this ambiguity by looking at words surrounding words.

A practical speech recognition system should have the following features:

* Speaker independent.
* Reasonable accuracy, i.e., above 80% recognition accuracy with connected words recognition systems.
* Accept spoken words in typical room conditions.
* Contains a large vocabulary base.
* Should be computationally efficient near to real-time.
* Easy training process.

There are two approaches followed in speech recognition, namely, the statistical pattern recognition approach and the acoustic phonetic approach.

3.3.1 *Speech Recognition by Statistical Pattern Recognition.*

The majority of the isolated-word recognition systems available commercially follow the model shown in Figure 7. End-point detection is the first process applied to the inputted utterance to locate the beginning and end points of the utterance. A technique based on energy thresholds and average

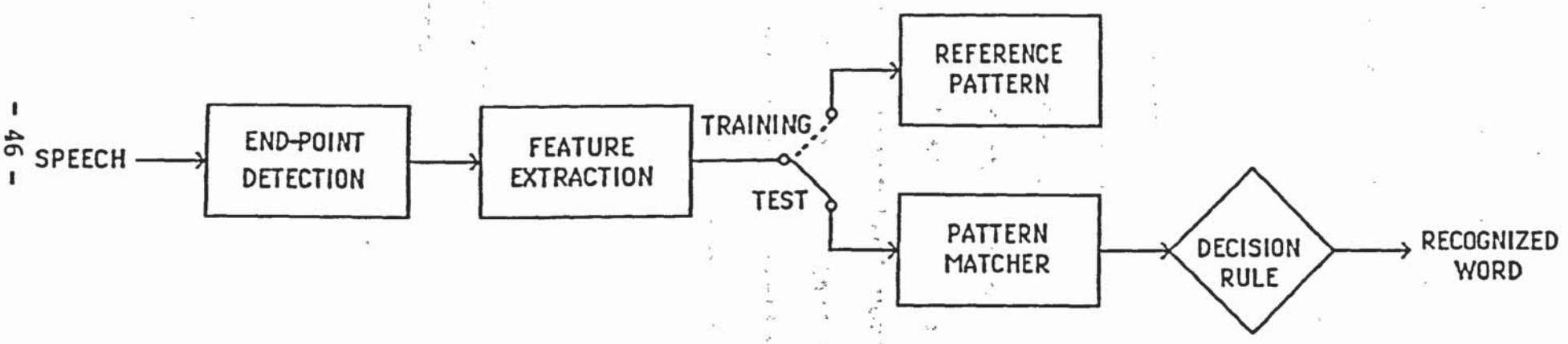


FIGURE 7

STATISTICAL PATTERN RECOGNITION.

zero-crossings is proposed by Rabiner and Sambur [24]. This technique performed well in locating end points of utterances, however when initial or final weak fricatives (/f/ and /th/) are involved, the performance of this technique was not so good. The second processing step is the feature extractor stage. In each time analysis window, a set of features (parameters) is extracted forming a feature vector. The third stage in processing is pattern matching, where the set of feature vectors is time aligned and pattern-matched with a set of reference templates. The output of this stage is a similarity measure which is fed finally into a decision-rule stage which selects the recognized word based on the best pattern-matching score. Similarity measurements are achieved by employing a particular distance measurement technique appropriate with the type of feature being used. Distance measures used in speech recognition are:

* Normalized (weighted) Euclidean distance.
* Euclidean distance/Correlation distance.
* Mahalanobis distance.

The type of features normally used in speech recognition systems are [23]:

Time Domain

* Average energy or amplitude.
* Average zero-crossings.

* Autocorrelation functions.
* Partial correlation coefficients (PARCOR).
* Linear prediction coding coefficients (LPC).
* Voiced/unvoiced indications.
* Pitch Values.

Frequency Domain

* Fourier spectrum coefficients.
* Cepstrum coefficients.
* Formant values.

Isolated-word recognition systems based on zero-crossing features are available commercially and the advantages of these systems is their hardware simplicity and real-time computation. The function of the time-alignment step is to find an alignment function which maps the reference pattern onto the corresponding parts of the test pattern. The criterion of the mapping function is such that it minimizes a measure of distance between the two patterns. Simple linear scaling of the time axis between the test and reference patterns is not a successful procedure for speech recognition and a better one is a non-linear time warp optimized for every pair of utterances. The optimum warping is derived by a dynamic programming procedure called "Dynamic Time Warping" (DTW) which is the technique

used to align optimally in time the test and reference patterns and to find the optimal distance measures associated with the optimal warping path. The main disadvantage of this technique is its heavy and complex computational requirement.

An alternative to storing word templates and matching them with the input speech patterns is by modelling the speech production process by a finite-state system model known as Hidden Markov model HMM, [25]. Each state of the model is capable of generating a finite number of possible outputs. In generating a word, the system passes from one state to another, each state emitting an output, until the entire word has been generated. Thus the finite-state model specifies for each word a sequence of states. Since normally each word is represented as a sequence of features which are extracted at regular time intervals (frames), one can associate each state with one or several frames. To allow for pronunciation time variations, the model can either stay in one state for several successive time frames, (i.e., loop), or can by-pass to the next state. The behaviour of the model is assumed to be of a stochastic nature, i.e., system operation is governed by a set of probabilities and the sequence of past events (states) does not affect the current state output of the model. An observer sees only the

observations generated by the model, and not the "hidden" states, thus the title "hidden" Markov Modelling. Transition from one state to another is governed by a known probability of occurrence. Similarly, emitting an observation from several possibilities is also governed by a known probability.

In the recognition mode, the finite-state model which best matches the given input pattern sequence is chosen. This model has the highest probability of generating the given observation sequence. Practical HMM models used for speech recognition use a five-state model of left-to-right transition as shown in Figure 8. The computational requirements for building up HMM models is the only disadvantage of this approach as compared to DTW. HMM based word recognizers have performed better than most DTW based recognizers [26].

3.3.2 *Acoustic Phonetic Approach (Recognition).*

Any language contains a set of mutually exclusive speech sounds called phonemes. These phonemes are assumed to have unique articulatory and acoustic correlates. Recognition of these phonemes from the acoustic signal is very difficult because the acoustic properties of a phoneme change as a function of the immediate phonetic environment, i.e. coarticulation, which is seen in the acoustic signal

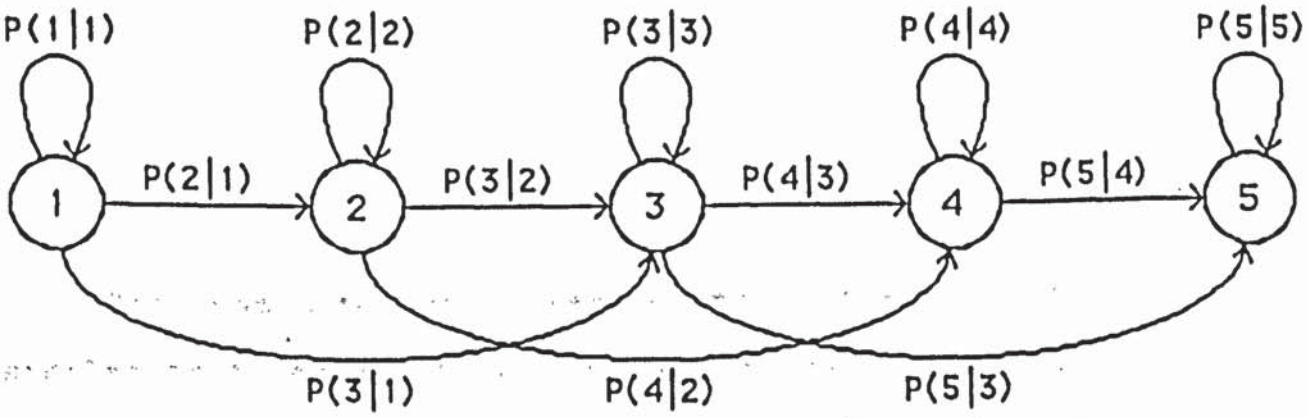


FIGURE 8

A LEFT-TO-RIGHT HIDDEN MARKOV MODEL.

as smearing of segmental boundaries [27]. Phonetic recognition strategies include stages of higher-level linguistic processing i.e., syntax and semantics. The basic concepts of this strategy are shown in Figure 9. The output of the acoustic analyzer is a set of parameters (features). The parameters are fed to the labelling and classification stage which generates all the possible words. Higher linguistic processing, i.e. syntax and semantics, identifies the correct word based on syntactic structure and semantic value.

The characteristic parameters extracted are similar to those mentioned before for the statistical pattern recognition approach. The advantage of phonetic recognition is that the total number of phonemes are small, hence a very large recognition vocabulary can be obtained. The disadvantage of this recognition approach is the large context-dependence of phonemes and the complexity of accurate algorithms for segmentation, labelling, and classification purposes.

3.4 *Arabic Speech Synthesis and Recognition*

Work on Arabic speech processing has been reported by the following Institutions:

* LEESA, Faculte Des Sciences [21].
* Scientific Studies and Research Center, Syria [28].
* University of Riyadh, Saudi Arabia [29].
* University of Cairo, Egypt [30].

FIGURE 9

ACOUSTIC PHONETIC RECOGNIZER.

* University of Baghdad, Iraq [31].

* University of Swansea, U.K. [32].

* Kuwait Scientific Center, Kuwait [33].

Arabic text-to-speech synthesis systems have been realized and reported by A. Mouradi [21] and M. Mrayati [28]. Both systems follow the same synthesis strategy where the diphones-synthesis method is used. The size of the diphones dictionary is in the order of 1000-1500. Both systems employ an orthographic-to-phonetic conversion module. The input to this module is the standard Arabic characters plus the diacritic marks (vowel signs).

Facilities to allow input of prosodic information in the orthographic-to-phonetic conversion program are reported by Mrayati but not Mouradi. The output of the orthographic-to-phonetic conversion program is a sequence of diphones and prosodic information. Both systems were implemented around digital research computer technologies (PDP-11/34 and VAX 11/780), and the diphone dictionaries are coded in LPC formats. Mouradi reported how the diphones were collected, i.e. from nonsense trisyllabic words. This is very important in order to allow for all possible contexts. No evaluations of these systems were reported, although both of them speak of intelligible speech output. Mouradi's system did not produce the speech quality desired for stop geminated consonants because his

definition of gemination is repetition of the consonant i.e., increased duration only. This definition did not seem to be working adequately for Arabic stop consonants.

Another Arabic text-to-speech system based on Arabic syllables as the units for speech stored in the dictionary is currently being developed at the Kuwait Scientific Center [33].

Attempts to synthesize Arabic speech using English-phoneme-based synthesizers have been reported by Mandurah [29]. Mandurah used the English-phoneme-based synthesizer made by Votrax Company i.e., SC-01 chip. This chip is capable of generating 64 English phonemes and allophones grouped into 25 consonant sounds, 20 basic vowel sounds, 16 durational vowel-allophone sounds, and 3 no-sound phonemes (pauses). Eighteen of the basic Arabic consonants and the Arabic vowels are generated from equivalence with the English phonemes, the rest of the Arabic consonants are simulated by a combination of sequences of other English phonemes. Speech output quality obtained from adopting an English phoneme synthesizer is reasonable, but not high enough for applications.

Arabic speech recognition has been reported by Cairo University [30]. They reported their experience with five isolated-word recognition systems. The approach followed was that of

statistical pattern recognition. They have experimented with end-point detection and feature selections. They evaluated the success rate of recognition against the different features i.e., normalized autocorrrelation coefficients, linear prediction coefficients (LPC), and cepstrum coefficients.

Arabic digit recognition systems have also been reported by W. Abdulla [31] and W. Mahmoud [32]. Abdulla reported a real-time Arabic digit recognition system based on analysis and knowledge of the zero-crossing rate as the main feature vector. He implemented his system with simple hardware interfaced with a microprocessor. The recognition accuracy achieved was 97% for a single speaker. Mahmoud reported a similar recognition strategy, except that his feature vectors were the sound intensity measurements in eight frequency bands. The mean recognition accuracy achieved by five speakers was 85%.

3.5 *Speech Synthesis and Recognition Applications*

The need for man-machine communication is very strong, therefore voice communication by and with computers has found many applications in modern technology. Many commercial systems are sold today serving various fields such as telecommunication, business and medical. The following speech applications are typical of ones found today:

* Realization of text-to-speech systems.
* Voice response systems.
* Phonetic typewriter.
* Voice activated systems (robotics).
* Airline reservation systems.

Man-machine speech communication with and by computers provides the user with extra physical mobility.

CHAPTER 4

Chapter 4

PC-BASED SPEECH PROCESSING SYSTEM

4.1 Introduction

Researchers working in the field of speech processing are faced with the need for adequate facilities in order to digitize, process, analyze, and store speech. Purchase of such facilities can be quite expensive, but fortunately with the advent of ever-increasing computational capabilities of personal computers, such systems and facilities can be built around personal computers which can easily be connected to mainframes through proper interfaces, hence capitalizing on the computational power of the mainframe. The development of a system for acoustic-phonetic analysis, synthesis and recognition of speech is reported here. The system is being developed to serve as an experimental tool for Arabic speech processing. The system is capable of digitizing, processing, analyzing and storing speech. The system is designed to be flexible and user-friendly.

The speech sampling rate (digitizing frequency) is determined by the bandwidth of the signal to be digitized (Nyquist criteria). For telephone quality speech, sampling rates of the order of 7-8 kHz are

adequate since the bandwidth is in the region of 3.4 kHz. For the system developed, the speech signal is band limited by an antiallaising low-pass filter to 4 kHz, with a speech sampling frequency of 10 kHz.

4.2 System Hardware Architecture

The system consists of a speech data acquisition subsystem and an IBM PC-AT with the following configuration:

* IBM PC-AT with 512 KB RAM.
* Enhanced Graphic Adaptor Card.
* High Resolution Colour Monitor.
* Co-math Processor 80287.
* 30 MB Hard Disk.
* DT2801 A/D Card from Data Translation, Inc. [34].

The speech data acquisition subsystem consists of the following components:

1. A unidirectional low noise dynamic microphone (Texas Instruments) is used. The frequency response of the microphone extends from 40 Hz to 20 kHz.
2. A low noise pre-amplifier is used to amplify the signal level from the microphone input. The amplified speech signal varies between ± 5 volts, which is the required input for the 12 bit linear A/D converter. The full range of

the A/D converter is fully utilized, thus minimizing the quantization error.

3. A low-pass filter is used to band-limit the speech spectrum to 4 kHz. This is necessary in order to prevent allaising distortion. This occurs at frequencies above half the sampling frequency, which is set to 10 kHz by the DT2801 board [34].

The MC145414 switched capacitor filter IC is used in the implementation of the low pass unity gain filter. The MC145414 is a 5-pole elliptic low-pass filter with tunable passband edges from 1.25 kHz to 10 kHz selected by a clock frequency that can be varied from 50 kHz to 400 kHz. Appendix B gives the details of the MC145414 chip with its low-pass filter frequency response characteristics.

4. A 12-bit linear A/D converter is used to digitize the speech pressure waves. The A/D converter is part of the DT2801 board [34] which is housed inside the IBM PC-AT. The dynamic range of speech sounds is very large, i.e., fricative sounds are very soft and weak while vowels are much more intense. Therefore, to encompass the entire dynamic range, a good resolution A/D converter is required. The 12-bit A/D converter used provides a resolution of 4096 levels of digitizations.

A block diagram of the whole system hardware architecture is shown in Figure 10.

4.3 *Speech Input and Storage*

The source of speech input can either be direct from the microphone or from a tape recorder. 12-bit encoded speech samples are acquired at a rate of 10 k samples/sec., hence 20,000 bytes of memory storage is filled in one second. The temporary storage used has a 64 K RAM capacity. Thus, a maximum of up to 3 seconds of speech messages can be stored. Speech messages are stored in hard disk files for later processing.

4.4 *Speech Processing and System Functions*

Speech signals sampled during recording sessions embody complex information that reflect the different articulation of sound units and their transitions from one sound unit to the other. Digital signal processing became increasingly important in a number of scientific and technical areas. One of the major applications of digital signal processing is in speech processing. Speech is assumed to be a time-invariant process over a short time interval (10 msec. duration), hence, short time analysis (linear and non-linear systems theory) can be applied.

The system is designed to be menu driven where a menu comprising the basic system functions namely, speech input, speech playback and speech analysis, appear on the IBM PC-AT enhanced colour monitor. The

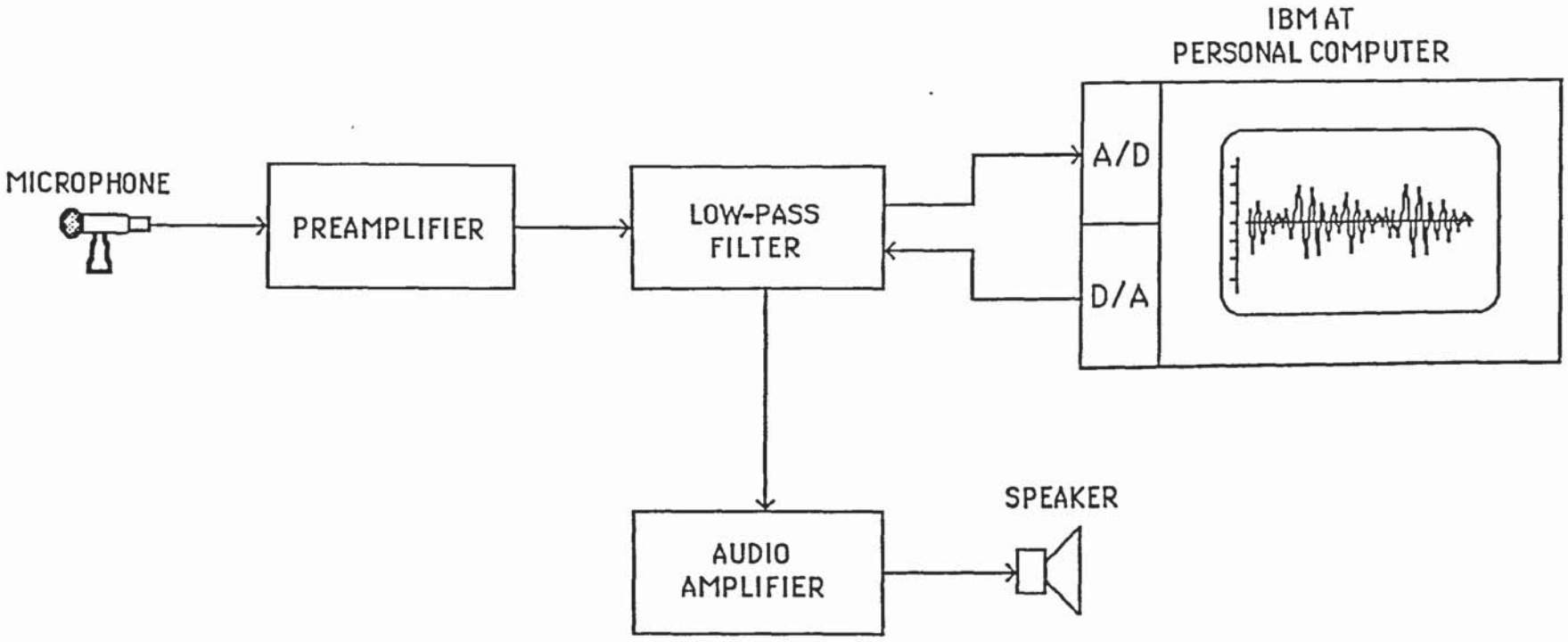


FIGURE 10

SYSTEM ARCHITECTURE BLOCK DIAGRAM.

entire software is made of assembly programs and PASCAL routines. Speech digitization, storage and playback is done through the assembly programs which are assembled and linked with the compiled PASCAL routines. Speech processing is carried out on frame sized speech segments which are the result of multiplying the speech samples by a sliding window sequence w(n). w(n) is a finite-length sequence that is non-zero for N samples and zero otherwise. A Hamming Window of size N=256 is used because it gives much greater attenuation outside the passband than the rectangular window [35]. The size of the window is important and, depending on the application,it should lie between 100 to 300 samples. The choice of 256 is made because it is a power of two for the FFT computation and also because it embraces more than one pitch period. During the analysis session one can select several functions such as:

* Speech play back.
* Utterance waveform plotting.
* LPC analysis and spectrum plotting.
* Absolute energy computation and zero-crossing analysis.
* Autocorrelation analysis.
* Discrete Fourier computation and plotting.
* Cepstrum analysis.

4.5 *Speech Editing*

Graphic facilities of the IBM PC-AT are exploited to enable plotting of speech waveforms of a whole utterance or part of it by either partitioning with cursor movements or selecting particular frames. Figure 11 shows the entire speech waveform of the Arabic utterance /jaʃkuru:n/. Analysis and waveform synthesis can be carried out on selected portions of speech. Arabic word, syllable, diphone and phoneme boundaries can be detected and identified using either of two methods:

1. Via visual inspection of the speech waveform and repeated speech playback activities.
2. Via computing and displaying the energy contour, profile which generally reflects the Arabic syllabic contents of the spoken utterance.

Hence an inventory of Arabic syllables, diphone and phonemes units can be built, thus permitting experimentation with Arabic speech synthesis techniques such as the simple concatenations of these synthesis units.

4.6 *Short Time Speech Analysis*

The characteristic speech properties are assumed to be constant over finite time durations typically from 10 to 30 msec. The change in temporal behavior of speech samples over frame lengths of 100 to 300

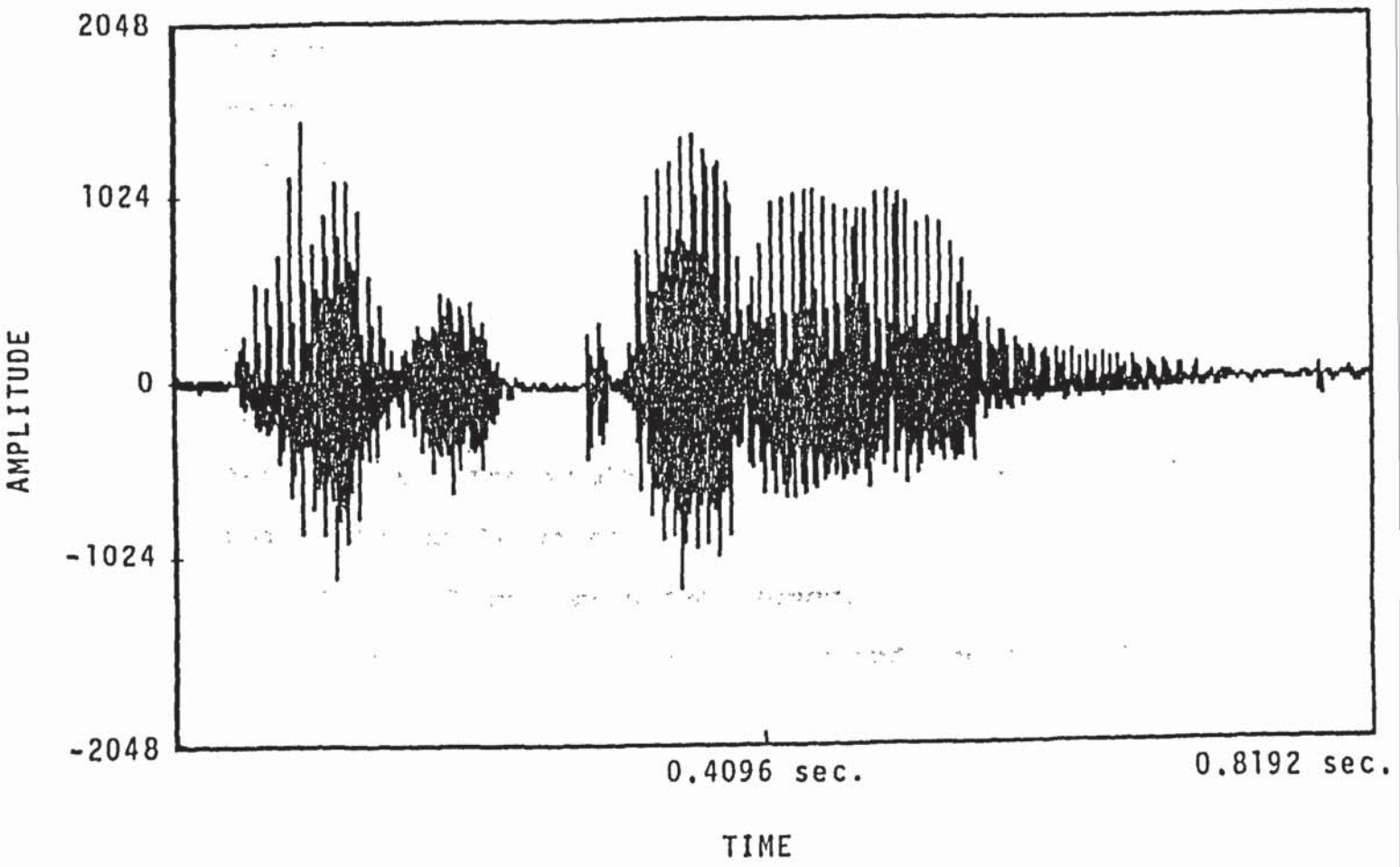


FIGURE 11

PRESSURE WAVEFORM OF THE SPEECH UTTERANCE /jaʃkuru:n/.

speech samples is very slow, hence "short time" speech processing analysis techniques can be applied in the time and frequency domains. Speech frames are obtained by multiplying the train of speech samples by an appropriate window function such as the Hamming Window. A tutorial review of the parametric representations of speech signals is reported by Rabiner and Schafer [35]. In the analysis of the time domain and frequency domain methods used in speech processing that follows, we shall let x(n) denote the speech sample at unit time n, $0 \leq n \leq N-1$, where N is equal to the frame size. We also let w(n) denote a sliding window function of size equal to N positioned at location n-t.

4.6.1 *Time Domain Techniques.*

Time domain speech processing methods deal directly with the speech waveform signals. They are favoured because of their simplicity in implementation. The following are examples of time domain methods used in speech processing.

1. ***Energy***

   The amplitude of the speech samples varies according to the type of speech sound and the suprasegmental factors such as pitch and stress. The short-time energy E_t is computed according to the equation below:

$$E_t = \sum_{n=t}^{t+N-1} \{x(n)\ w(n-t)\}^2 \qquad (2)$$

The shape of w(n) is not critical for energy computation so a rectangular window could be used [36]. Because of the squaring operation, the energy can be sensitive to large signal levels. For this reason and for faster computation, an average magnitude term as defined in equation 3 is used.

$$M_t = \sum_{n=t}^{t+N-1} |x(n)|\ w(n-t) \qquad (3)$$

The energy parameter is very useful in speech segmentation and end-point detection [24]. This parameter is used later on for the Arabic syllabic segmentation algorithm.

2. *Zero-Crossing*

The zero-crossing rate (ZCR) is obtained by counting the number of times the signal changes sign during one frame interval as given in the equation below:

$$ZCR_t = \sum_{n=t}^{t+N-1} |Sgn\{x(n)\}-Sgn\{x(n-1)\}|\ w(n-t) \qquad (4)$$

ZCR gives a rough estimate of the frequency content of the speech signals, i.e., a sinusoidal signal of frequency F gives an

average ZCR of 2F/sec. ZCR has been used by Rabiner [24] to improve the end-point detection algorithm. It has also been used to distinguish between voiced and unvoiced sounds, i.e., mean ZCR for voiced sounds is less than 1500/sec. while for unvoiced sound it is more than 5000/sec. [36].

3. ***Autocorrelation Function R(m)***

The short-time autocorrelation function R(m) is computed according to the equation below:

$$R_t(m) = \sum_{n=t}^{t+N-1-m} x(n) \times w(n-t) \times (n+m)\, w(n+m-t) \quad (5)$$

R(m) is very useful in determining the periodic behaviour of the speech signals. It is sometimes used to give an estimate of pitch values. R(0) is an indication of the energy content in a speech frame, therefore it is very useful for normalization purposes.

4. ***Linear Prediction (LP)***

Linear prediction is closely related to the basic model of speech production in which the speech signal is modelled as the output of a linear, time varying system excited by either quasi-periodic pulses for voiced sounds/or white random noise for unvoiced sounds. LP is

a very important technique for speech analysis, synthesis and recognition, as the basic speech parameters, e.g., pitch, formants, spectra and vocal tract area function can be easily estimated with reasonable accuracy. The basic idea behind LPC is that at any instant of time, the speech sample can be approximated as a linear combination of P past samples. The size of P varies between 8 and 14. A thorough treatment of this technique is given by Makhoul [12].

4.6.2 Frequency Domain Techniques.

Spectral representations of the speech signals display characteristics that are not evident in the time speech waveform. It is therefore an important parametric representation that is useful for speech synthesis and recognition applications. Two important considerations affect the shape of the speech spectrum, namely the type of window used, i.e., Hamming or Rectangular, and pre-emphasis. The shape of the window function w(n) is important because multiplication of the speech samples by w(n) in the time domain means convolution of the Fourier Transforms of both w(n) and the speech samples in the frequency domain. Therefore, the Fourier Transform of w(n) is required to have little attenuation inside the passband to prevent smearing of the speech

spectrum and to have greater attenuation outside the passband. In this respect the Hamming Window is the most suitable. Pre-emphasis is applied to speech samples in order to compensate the 6-12 dB/octave fall in the spectrum of voiced sounds. The effect of a +6 dB/octave lift in the speech spectrum can be achieved digitally by differencing the input as given in the equation below:

$$y(n) = x(n) - ax(n-1) \qquad (6)$$

where,

y(n) is the output speech signal at unit time n

a is a coefficient of value between 0.9 and 1

Examples of short-time frequency-domain methods are:

1. ***Discrete Fourier Transform (DFT)***

   The spectral representation of speech signals is one of the most important parameters in speech analysis as it reflects the time-varying properties of the speech waveforms over finite time intervals (10-20 msec.). The fundamental equation for the computation of the Discrete Short-Time Spectrum of x(n) is defined as:

$$X_t(k) = \sum_{n=t}^{t+N-1} x\,(n)\; w(n-t)\; e^{-j2\pi kn/N} \qquad (7)$$

with k=0, 1, ... N-1

and the Inverse Discrete Fourier Transform is defined as:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} x\,(k)\; e^{+j2\pi kn/N} \quad n=0, \ldots N-1 \qquad (8)$$

Direct use of equation 7 for the computation of the speech spectrum is not convenient, so the Fast Fourier Transform (FFT) is used instead. The FFT is a set of highly efficient algorithms for evaluating the Discrete Fourier Transform (DFT) expressions [37].

2. ***Cepstrum Analysis***

This technique models the speech production process as the convolution of an excitation function (either white random noise or a quasi-periodic pulse train) with the vocal tract impulse response (homomorphic system). Homomorphic system analysis techniques are very useful in separating signals that have been combined by convolution. The cepstrum function, C(n), is the Inverse Discrete Fourier Transform of the logarithm of the magnitude of the Fourier Transform of the speech signals, as

shown in Figure 12. The low order cepstrum coefficients represent the slowly varying components due to vocal tract transmission while the high order cepstrum coefficients represent the periodic components due to the periodic excitation. The rapidly varying periodic component of the log-magnitude manifests itself in the strong peak at a time equal to the period of the input speech signals.

4.7 Software System Implementation

All software routines responsible for the speech analysis sessions are coded, and compiled in Turbo PASCAL. These routines are shown in Appendix C.

4.8 Example of an Analysis Session

As an example, the spectrum of the Arabic vowel /a/ was generated using a 256 point FFT algorithm and is shown as a solid line in Figure 13. The same spectrum was also estimated from LPC coefficients [12], and is shown as the dotted line in Figure 13. Formant frequencies are clearly in evidence in both plots, however, the dotted plot has fewer extraneous peaks. This is because the order of the LP model was chosen to be 14, hence at most 7 resonance peaks could occur.

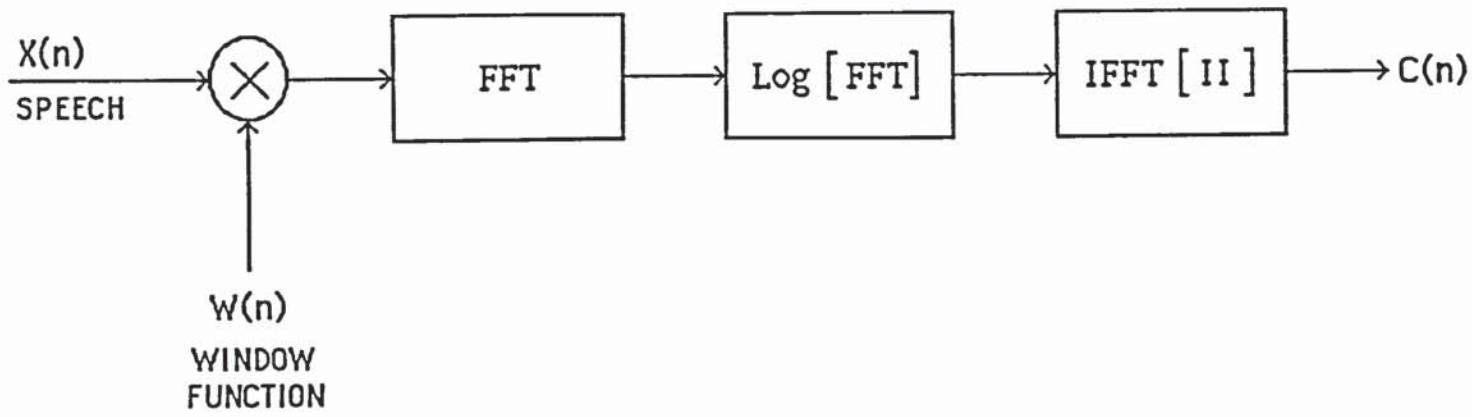


FIGURE 12

CEPSTRUM ANALYSIS.

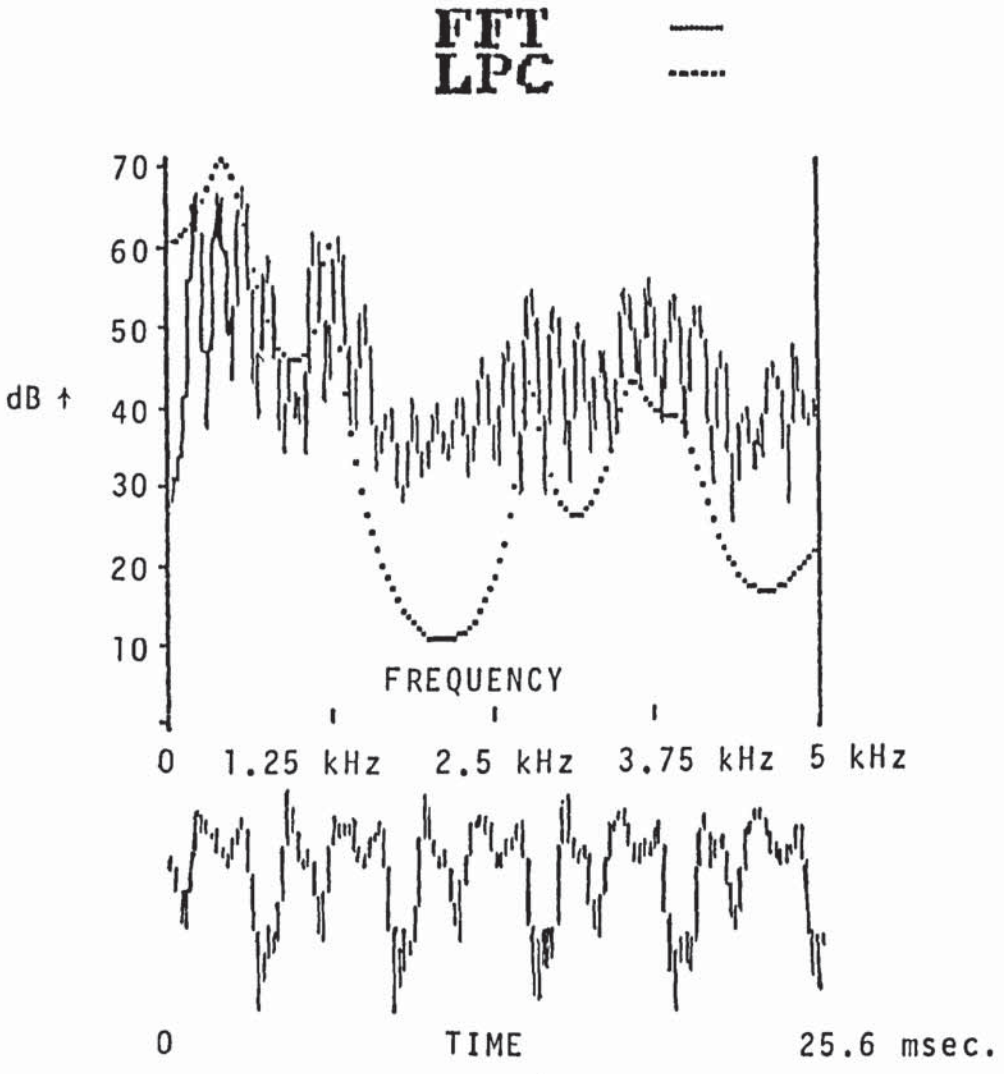


FIGURE 13

FFT AND LPC SPECTRUM OF ARABIC VOWEL /a/.

4.9 Discussion of System Uses and Expansion

The developed PC-AT-based speech-processing system has proven to be a very useful tool for conducting research work in speech-processing, such as the following:

* Acoustic-phonetic speech analysis.
* Speech synthesis.
* Speech recognition.
* Studies of language prosodic features such as pitch and intonation (pitch contours).

System ease-of-use and modularity allow for system functions to be altered, changed and enhanced. As it stands now, the system works as a stand-alone unit. Hence, its performance in terms of processing functions and computational speed is limited. Therefore, the system functions and performance would be enhanced if the following features are added:

1. Connection to the mainframe. This facility will allow more speech files to be stored in the mainframe and most important is that the time-consuming speech-processing functions such as spectrum and spectrographic analysis can be carried out by the mainframe which has much more powerful computational power. It will also mean that speech-processing packages such

as the Interactive Laboratory System (ILS) [38] can be used.

2. Addition of dedicated signal processing hardware boards such as TI-speech [39] which houses the TMS 320 signal processing chip.

The system has been developed as part of this research work in Arabic speech processing with emphasis on an acoustic-phonetics strategy for Arabic speech recognition.

CHAPTER 5

Chapter 5

PROPERTIES OF ACOUSTICALLY SIMILAR ARABIC SOUNDS

5.1 *Introduction*

Sound spectrograms are the most common used instruments in speech research. They provide visible patterns which show the important information in sounds coming from a speaker's mouth at a particular moment. Spectrograms are three dimensional plots, the vertical axis is the frequency scale, the horizontal axis is the time scale and the third axis is the intensity scale where the points are plotted. Figure 14 shows a typical spectrogram plot for the Arabic utterance /jatamaʃa/. The dark bands appearing in the spectrogram running across the pattern, represent the formants (harmonics) of the vocal tract system. The first intensity bar on the low frequency axis represents a measure of the first formant F1, the second one represents F2 and so on.

Spectrograms can be obtained with analysis filters of bandwidths 45 Hz (narrow band) and 300 Hz (wide band). The selected filter governs the response resolution. Normally wide band spectrograms are used for sound analysis where the time resolution is very high. However, if one needs to examine sustained sounds and if more frequency resolution is

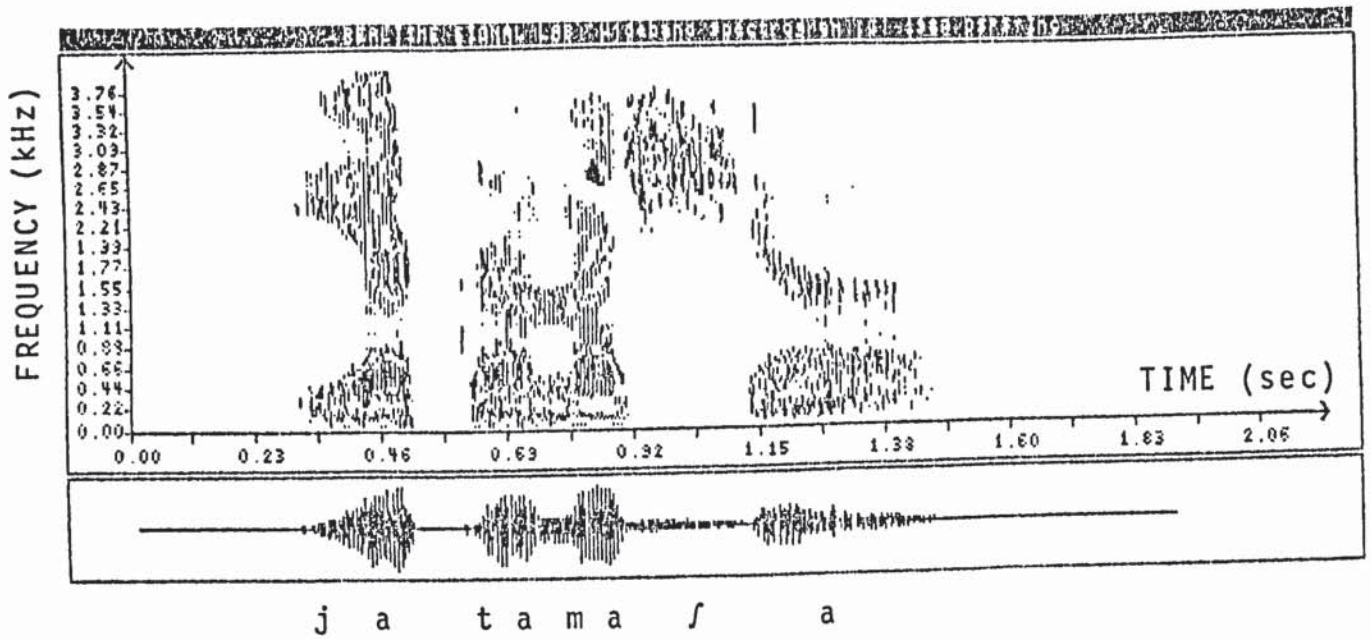


FIGURE 14

TYPICAL SPEECH SPECTROGRAM PLOT.

required then narrow band spectrograms are used, e.g., fundamental frequency analysis.

Spectrographic analysis is widely used in studies geared towards identifying speech cues related to speech perception. For example, the second formant transition in vowels has been found to be a very important acoustic cue for the identification of adjacent stops [40]. Other examples of speech cues include the durational and intensity characteristics as physical cues for the perception of linguistics stress [41].

The purpose of this investigation is to perform a sample study on Arabic similar sounds units. The study is aimed at examining the acoustical characteristics of such sound units under different Arabic vowel contexts, i.e., /a/, /i/ and /u/. The Arabic speech sounds used for this investigation include the pair of voiceless phonemes (/θ/, /f/) and the pair of voiced phonemes (/d/, /ḍ/).

5.2 *Equipment Used*

The system used to generate the speech spectrograms incorporated the following units:

1. An IBM PC/XT with 256K RAM.
2. Texas Instrument "TI-Speech" Board, [39].
3. TECMAR Graphics Master Card, [42].
4. Realtime Signal Lab. (RSL) Software, [43].
5. Sansui D-290 Tape Recorder.

All recordings were done in the normal office environment.

5.3 *Test Speech Materials*

The speech utterances used to perform this investigation consisted of syllables of the type /CVC/, where the pre-vocalic and post-vocalic sounds are the pairs of Arabic phonemes mentioned before. A total of 36 syllables uttered by three male speakers, were used.

5.4 *Acoustic Analysis*

The acoustical measurements carried out on each syllable were, total duration, duration of steady state portions, durations of transitions in and out of vocalic vicinities, formant values, energy values, and finally zero-crossing rates.

5.5 *Results and Discussions*

Spectrographic analysis of the Arabic vowel system was carried out first, the purpose was to determine the basic mean formant frequencies. Figures 15 to 20 show the spectrograms of the Arabic vowels /a/, /a:/, /i/, /i:/, /u/ and /u:/ uttered in isolation, i.e., started with an initial glottal stop. From these plots, we notice that Arabic has a well marked fourth resonance. Table 8 lists the mean formant and bandwidth frequencies of three male speakers.

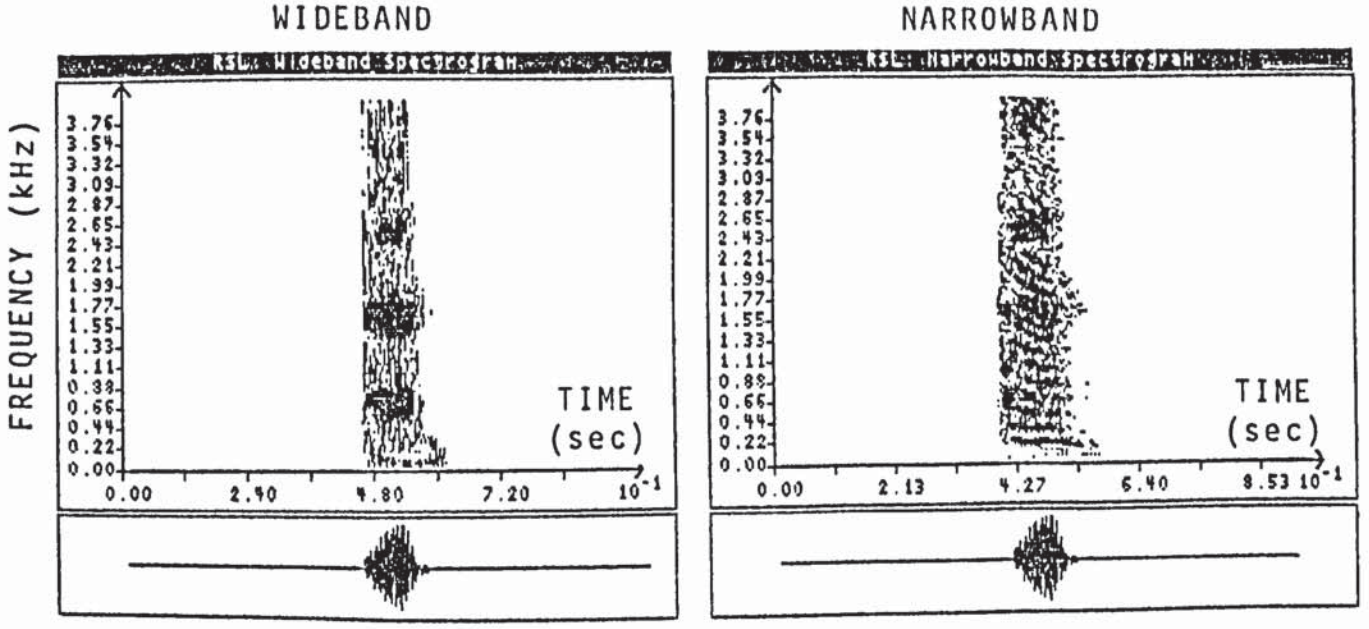


FIGURE 15

WIDE AND NARROW BAND SPECTROGRAM OF THE ARABIC VOWEL /a/.

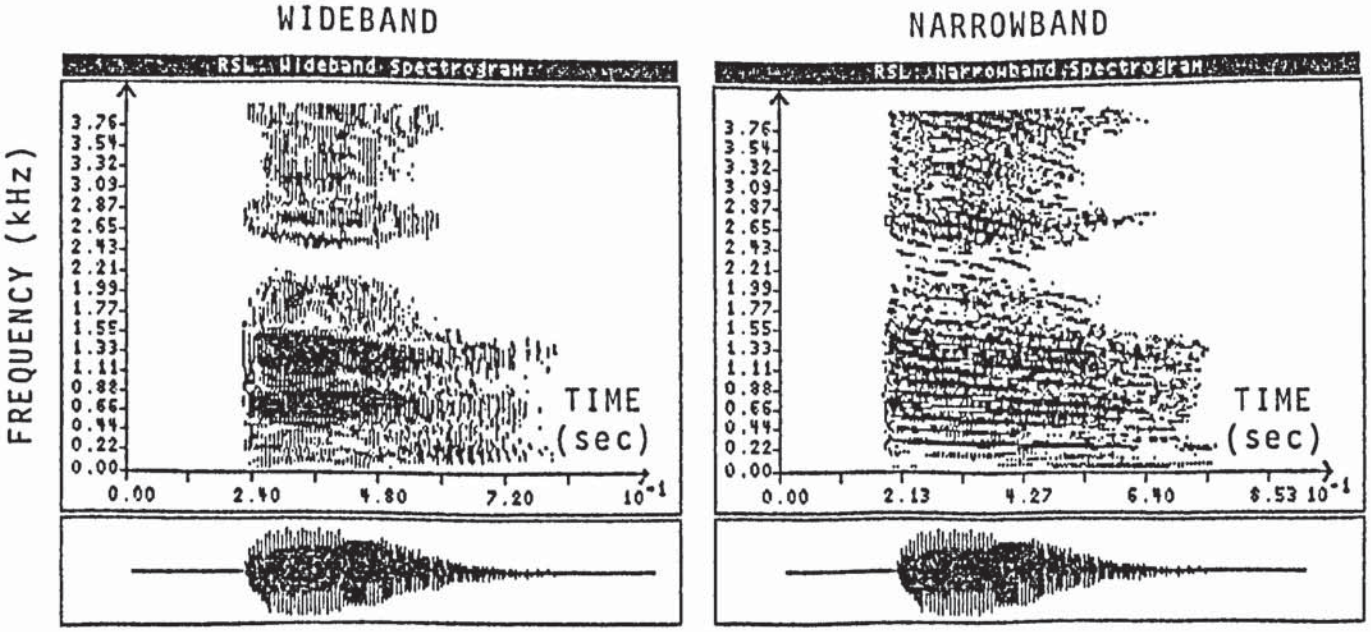


FIGURE 16

WIDE AND NARROW BAND SPECTROGRAM OF THE ARABIC VOWEL /a:/.

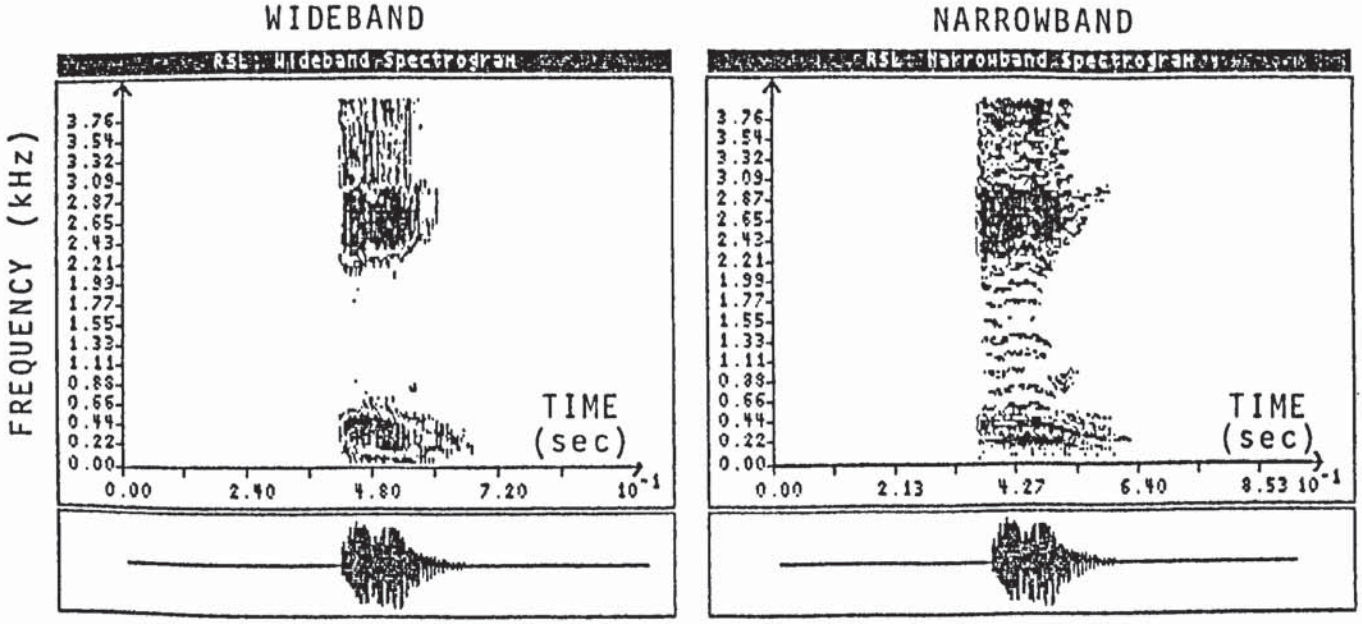


FIGURE 17

WIDE AND NARROW BAND SPECTROGRAM OF THE ARABIC VOWEL /i/.

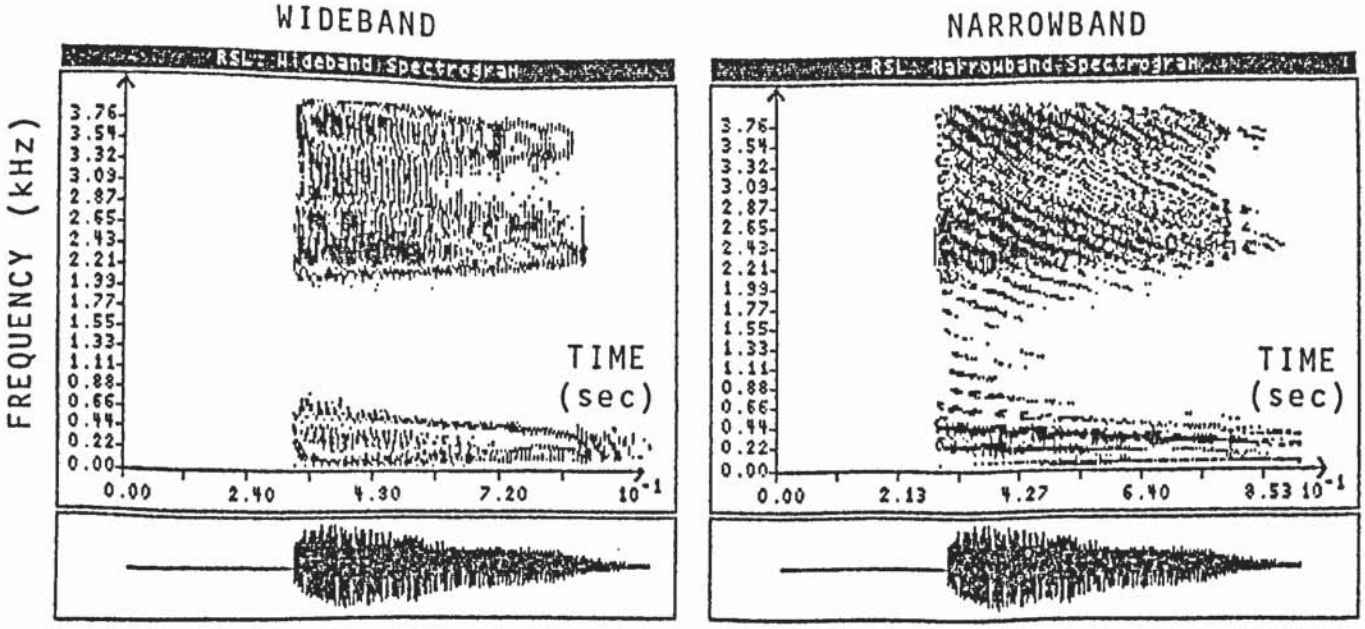


FIGURE 18

WIDE AND NARROW BAND SPECTROGRAM OF THE ARABIC VOWEL /i:/.

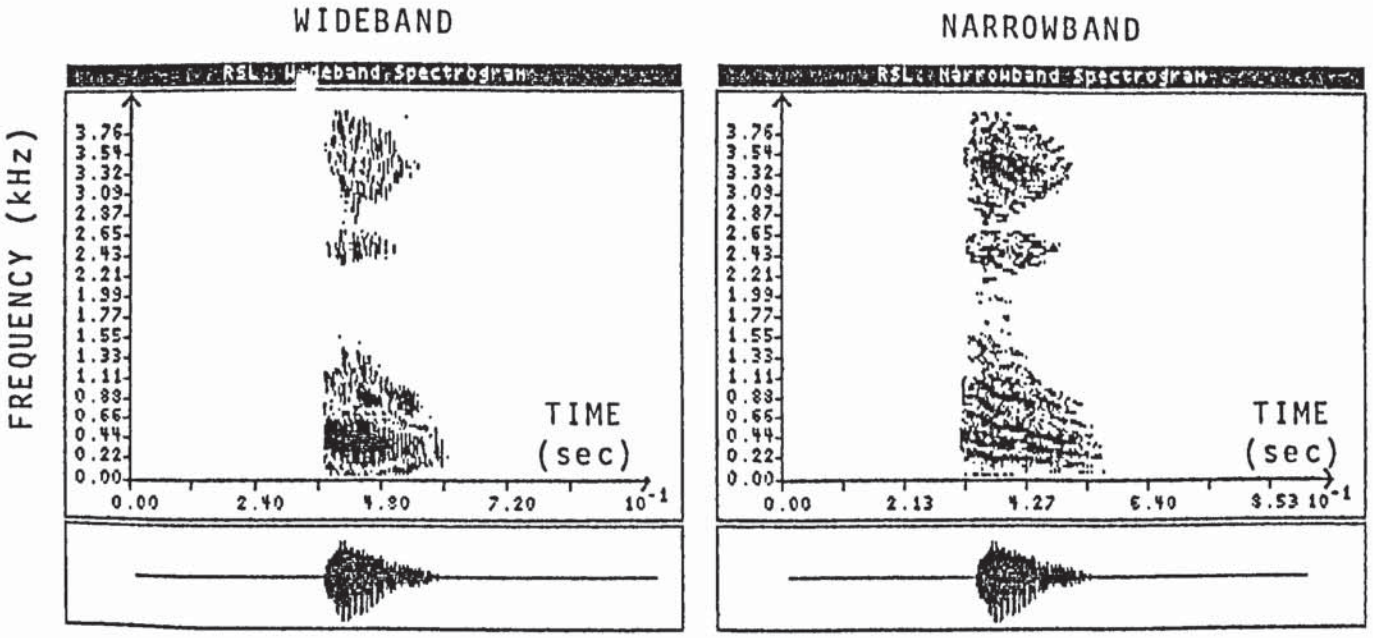


FIGURE 19

WIDE AND NARROW BAND SPECTROGRAM OF THE ARABIC VOWEL /u/.

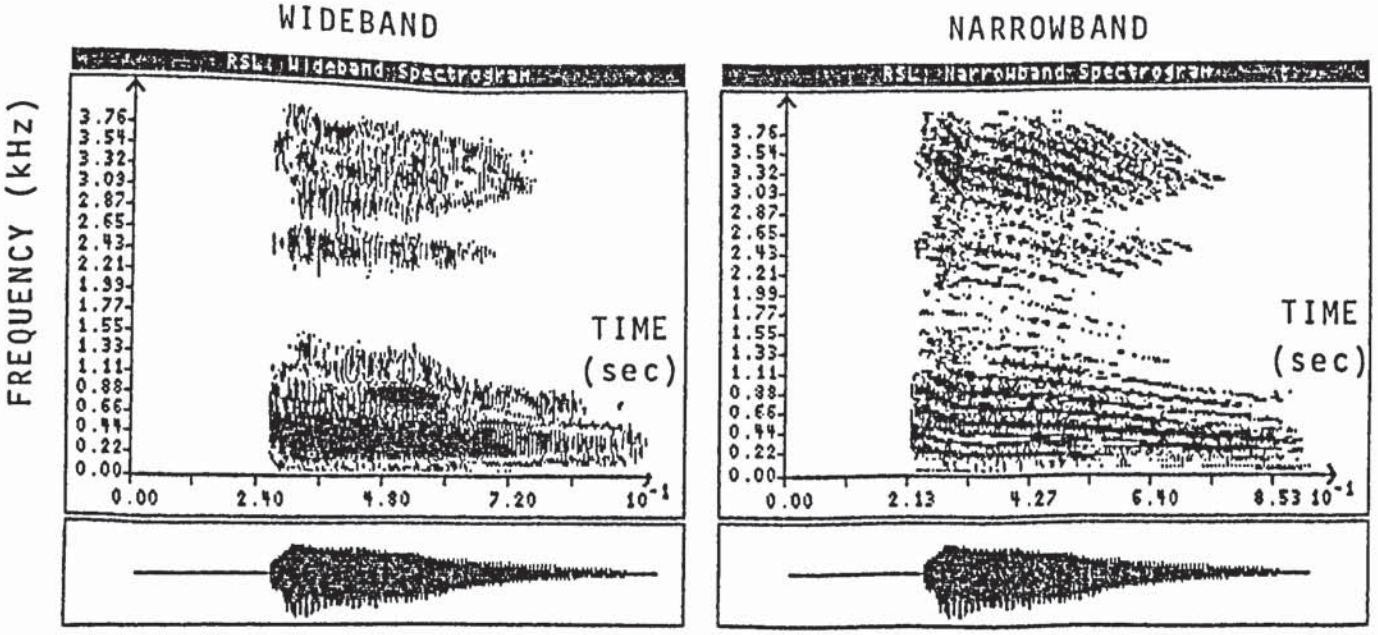


FIGURE 20

WIDE AND NARROW BAND SPECTROGRAM OF THE ARABIC VOWEL /u:/.

Arabic Vowels	Duration msec.	FORMANTS in Hz			BANDWIDTHSin Hz		
		F1	F2	F3	B1	B2	B3
a	130	660	1550	2650	50	62	100
a:	260	770	1330	2650	52	60	100
i	195	330	2430	2760	40	100	100
i:	390	330	2430	2760	40	100	100
u	180	330	880	2430	40	54	100
u:	360	330	880	2430	40	54	100

Table 8

Arabic Vowels Formants and Bandwidths Values.

The first two Arabic sounds /θ/ and /f/ represent a pair of voiceless fricatives, the place of articulation for /θ/ is interdental, and for /f/ is labiodental. Both sounds are characterized by having a spectrum similar to that generated by random noise, i.e., high zero-crossing rates and weak energies, however, phoneme θ has a slightly stronger energy than /f/. The measured steady state durations of both phonemes were about 70-90 msec., the durations of transitional phases into and out of vowel vicinities were also the same, and in the order of 20-30 msec. The noise formant values of phonemes /θ/ and /f/ start from 800 Hz and 550 Hz, respectively. Figures 21 and 22 show the extended spectrograms of the syllables /θaθ/ and /faf/, respectively.

There are no noticeable formant movements during the transition phases for both sounds, except for the first formant in the sound /θ/, where there is a marked dip from in its high noise value followed by a gradual increase until it reaches the steady state target value of the neighbouring vowel. This dip in F1 also coincides with the start of voicing. The rate of zero-crossing is initially very high. It sharply drops as the point of entry to the vowel is reached, after which voicing starts and the zero-crossing rate begins to pick up again. The

WIDEBAND

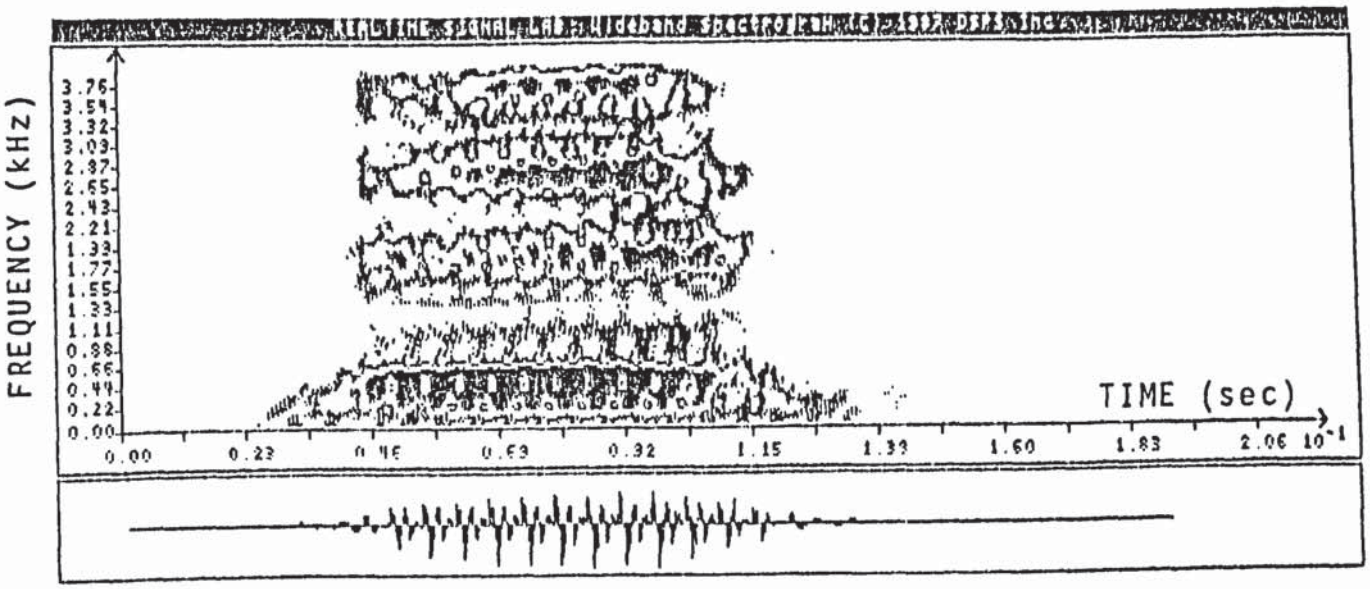


FIGURE 21

EXTENDED SPECTROGRAM OF THE ARABIC SYLLABLE /θaθ/.

WIDEBAND

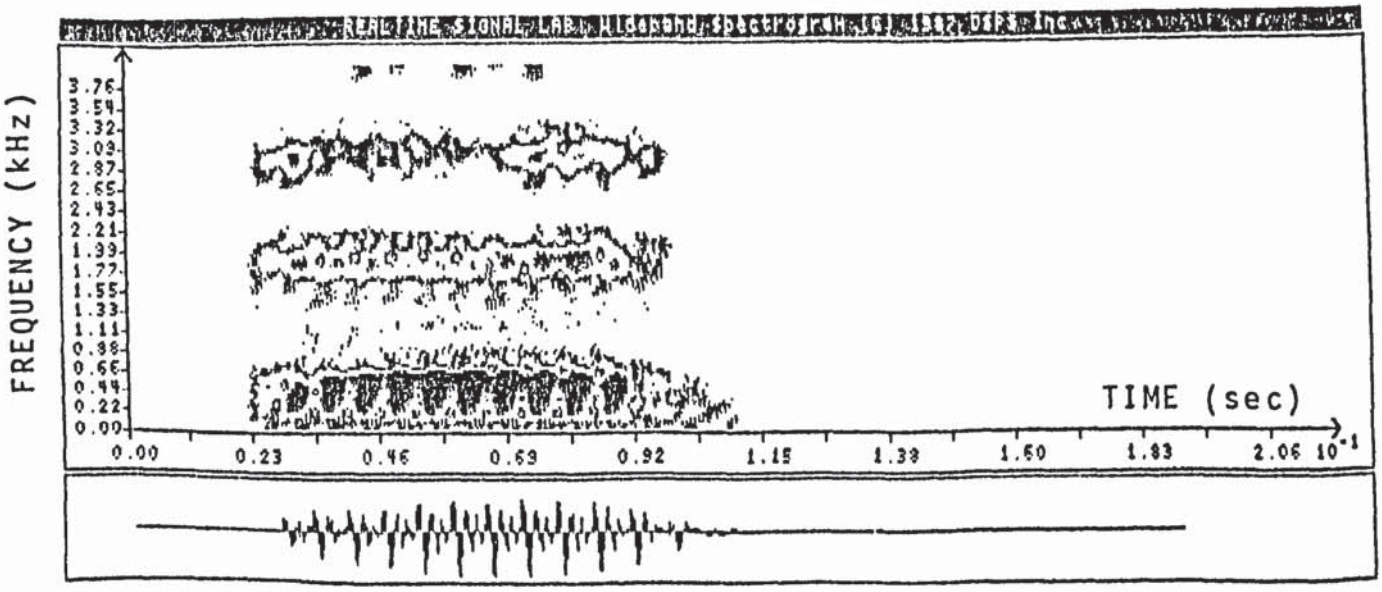


FIGURE 22

EXTENDED SPECTROGRAM OF THE ARABIC SYLLABLE /faf/.

minimum value of the zero-crossing rate also coincides with the dip and transitional movements of F1 for the consonant /θ/. The second formant value F2, of the Arabic vowel /i/ is affected by the phonemes /θ/ and /f/, i.e., it is slightly lowered to about 1000-2000 Hz from the normal steady state value of 2340 Hz.

The pair of Arabic speech sounds /d/ and /ḍ/ are classified as voiced dental stop consonants. Stop consonants are characterized by the formation of a closure along a point in the vocal tract system, causing air pressure to build up, and then sudden release is allowed. This manner of articulation is identified in the speech spectrograms by the presence of discontinuity (gaps) in the speech spectrum. The burst duration of /d/ and /ḍ/ are almost equal and is in the order of 15-20 msec. The voice on set time, (VOT) is also the same for both phonemes and is leading by 20-25 msecs. The measured mean formant values F1, F2 and F3 for /d/ are 250 Hz, 1700 Hz, 2900 Hz, respectively, and for /ḍ/ are 250 Hz, 1900 Hz and 3100 Hz respectively. Figures 23 and 24 show the extended spectrograms of the Arabic syllables /dad/ and /ḍaḍ/, respectively. Formant movements during the transitional phases are very well noticed especially the first and second formant, where for F1 there is an upward movement towards the vowel target

WIDEBAND

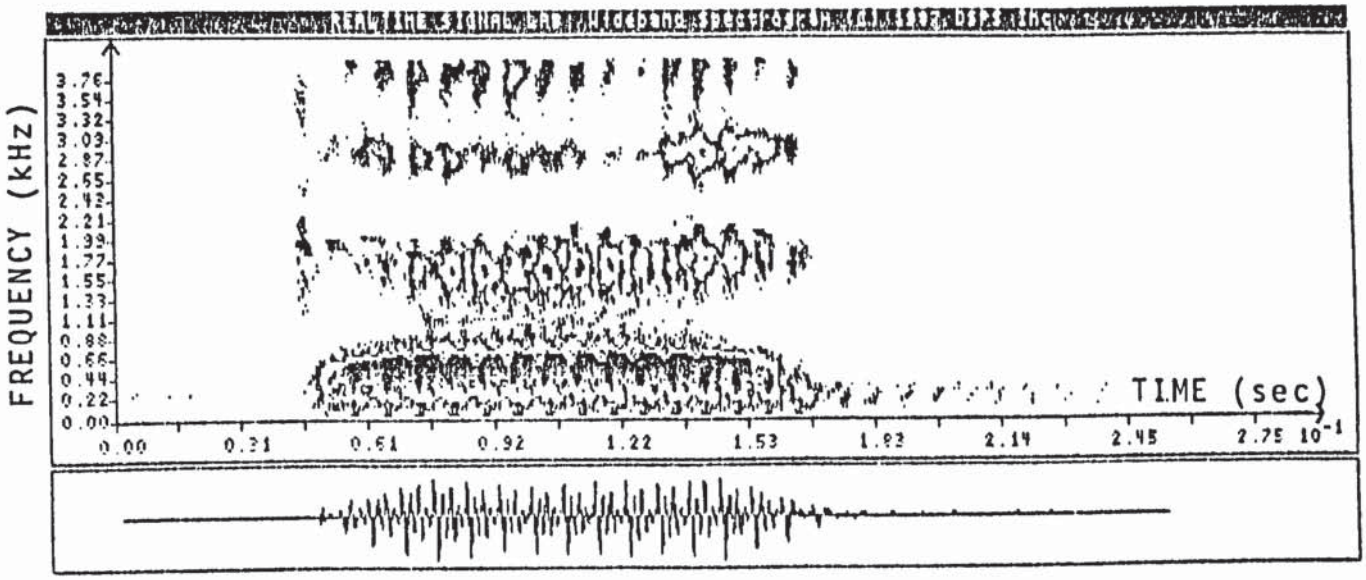


FIGURE 23

EXTENDED SPECTROGRAM OF THE ARABIC SYLLABLE /dad/.

WIDEBAND

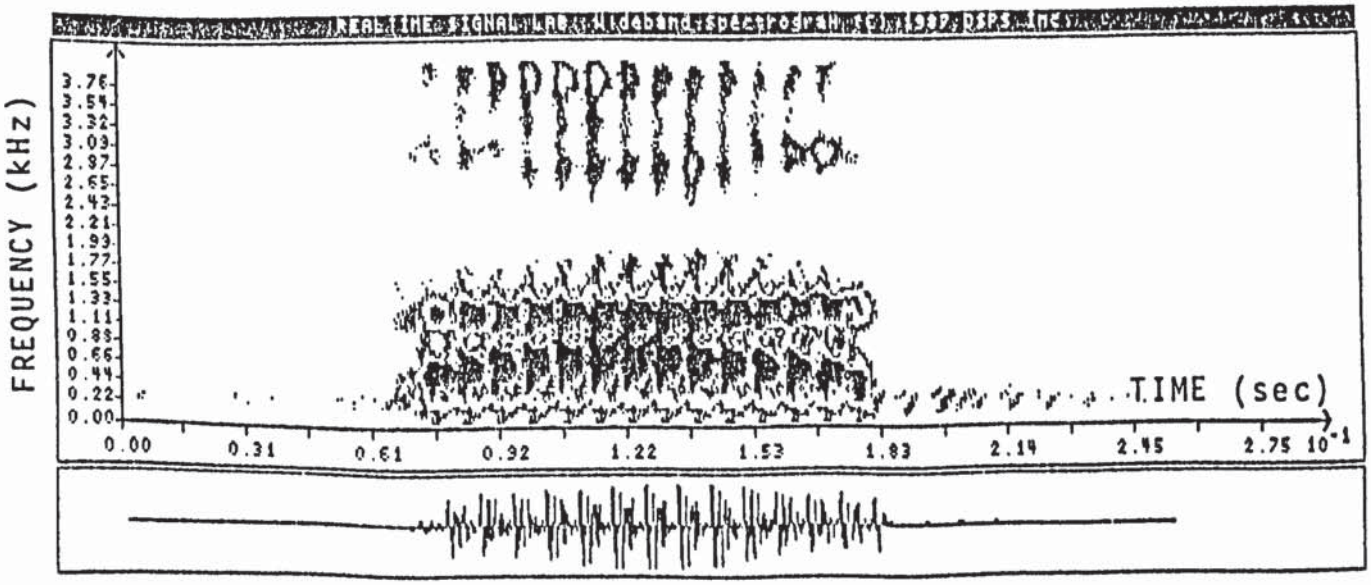


FIGURE 24

EXTENDED SPECTROGRAM OF THE ARABIC SYLLABLE /ɖaɖ/.

steady state value, while for F2 there is a downward movement towards the vowel steady state target value. Figure 25 shows a summary of F1 and F2 transitional behaviours of the stop consonants /d/ and /ḋ/ under the different Arabic vowel contexts.

The phoneme /d/ influences the onset value of F2 of Arabic vowels /u/ and /i/. For vowel /i/, F2 value is lowered while for /u/ F2 value is slightly raised. Similarly, phoneme /d/ influences the neighbouring vowel /a/ and /i/ by lowering their F2 onset value.

5.6 Conclusion

This study was concerned with the acoustical properties of similar Arabic speech sounds. Two pairs of Arabic consonants were studied, namely, (/θ/, /f/) and (/d/, /ḋ/). The acoustical measurements of the spoken CVC utterances was performed using spectrographic analysis. Typical acoustical measured properties are formant values and formant movements. The following main points were drawn from observations of the results:

1. For the pair of Arabic consonants /θ/ and /f/, a clear distinction between them is the steady rise of F1 during formant movement for the case of phoneme /θ/.
2. There is no appreciable measured acoustical property difference between /d/ and /ḋ/ except

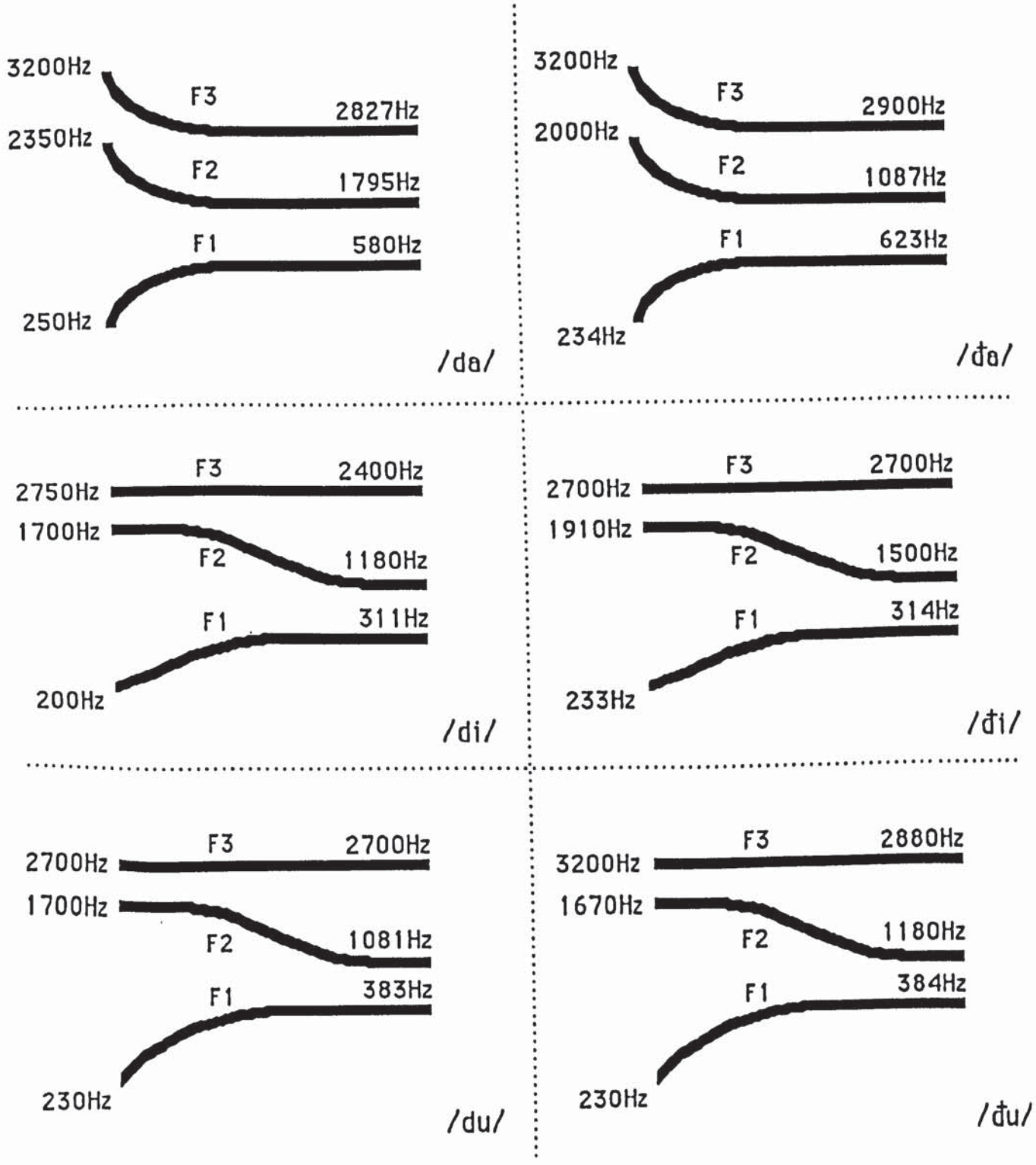


FIGURE 25

FORMANTS TRANSITIONS OF CONSONANTS
/d/ AND /ḍ/ BEFORE VARIOUS ARABIC VOWELS.

from the burst formant values and the influences /d/ and /ḍ/ makes on the onset values of F2 by the neighbouring vowels.

CHAPTER 6

Chapter 6

ARABIC SYLLABIC SEGMENTATION ALGORITHM

6.1 Concept of Segmentation

Speech is a complex acoustic signal (phonetic plus prosodic information mixed together) that results from the interaction of two independent functions, namely, the vocal tract system function and the excitation function. It is characterized by being of a continuant nature (hardly any pauses exist except at the end of words or phrases or while articulating stops) and of high data rate (10,000 samples per second assuming a sampling frequency of 10 kHz). It is therefore very advantageous to segment the steady stream of speech samples into minimal segments corresponding to minimal sound units (phonemes) in order to ease the computation of the characteristic speech parameters and to be able to group together acoustically homogenous segments. Because of speech variability and other problems such as noise, one to one correspondence between the acoustic events, represented by the speech signal and the corresponding phonemes, does not exist. It is therefore extremely difficult to automatically segment speech into phonemes by locating the beginning and end locations of each phoneme present in the utterance.

In practice, two methods of speech segmentation have been used, namely, segmentation into constant time frame intervals (e.g., 10 msec. duration), and segmentation into one of several broad-phonetic classes of segments based on the extraction of certain acoustic parameters.

Segmentation of speech signals on a time basis is simply accomplished by dividing the speech continuum into equal size time frames with typical duration of 10 msec. The obvious drawback of this approach is the large size of storage area required. Many of the speech recognition systems that have been reported in the literature follow this strategy of segmentation [44]. As an example, Reddy [44] reported a segmentation algorithm, where, first the speech waves are divided into a succession of minimal segments of 10 msec. in duration. Then these segments are added to adjacent ones based on an acoustic measure of similarity (intensity and zero-crossing levels) between the two adjacent segments. Similar segments are grouped together to form what Reddy called "sustained" segments whose duration are not less than 30 msec. The omitted segments are named "transitional" segments. Reddy associated the transitional segments with phoneme boundaries and the sustained segments as aiding towards phonetic transcription. Testing his system

with a single speaker and 30 different sentences, he reported satisfactory results in isolating phoneme boundaries (no actual figure was quoted).

Segmentation into one of several broad-phonetic classes of segments is achieved by extracting a number of parameters that are used to characterize different speech sounds. A number of parameters have been commonly used for broad-phonetic class characterization, e.g., the fundamental frequency Fo can be used to segment speech into voiced and unvoiced segments. Another example is the variations in intensity levels as a function of time for characterizing speech into vowel-like and non vowel-like sounds [45].

Benedetto [45] partitioned the speech continuum into four broad-phonetic class segments, namely, vowel-like, silence, fricatives, and other sounds. Each word in the vocabulary is represented as a series of combinations of these labelled segments. The dynamic time warping technique (DTW) is used to match the labelled segments with the set of stored reference segments. Benedetto's speech recognition system was tested using three speakers (one female and two males) and with a vocabulary of 30 long sentences. Recognition score of 80% accuracy was reported.

6.2 *Arabic Syllables as a Unit of Speech Recognition*

Syllables are important sound units for speech recognition, because coarticulation effects are already included in these units and also because they represent perceptual units. Syllables may be defined linguistically in terms of the inherent sonarity of each sound [5]. Peaks of sonarity generally correspond to peaks of syllabicity (syllable nucleus), but one cannot empirically extract sonarity information alone from the acoustic signal as other information exist in parallel such as prosody (stress and pitch). This mix-up of prosodic information and sonarity makes segmentation into syllables for the English language a rather difficult task to accomplish accurately. Arabic, on the other hand, has well defined rules governing grouping of phonemes into syllables, formation of syllables within a word and placement of stress on individual syllables within a word. It is thus inclined more towards simple decomposition into syllabic units.

6.3 *Syllabic Segmentation for the English Language*

The first developed automatic syllabic segmentation procedure is credited to Mermelstein [46]. The procedure involves the computation of the following two parameters:

1. A "Loudness" function which is obtained from the speech power spectrum by weighting frequencies below 500 Hz and above 4 kHz

according to a function that drops off at 12 dB/octave outside these frequencies. The Loudness function is smoothed by a 40 Hz low-pass filter.

2. The convex hull of the Loudness function is defined as the minimal-magnitude function that is monotonically non-decreasing from the start of the segment to its point of maximum loudness, and is monotonically non-increasing thereafter. In other words, the convex hull is the shape a piece of elastic would assume if stretched over the top of the Loudness function and anchored down at both ends, as shown in Figure 26.

The Loudness function is compared with its convex hull, and the point of maximum difference is taken to be a tentative syllable boundary. If the difference exceeds a given threshold, the segment is divided into two subsequences and the convex hull is recomputed, but this time anchored to the actual Loudness function at the tentative boundary. The points of maximum convex hull loudness difference are then selected as further tentative syllables boundaries. The same procedure is repeated until the convex hull loudness difference is below a given threshold ($< 4dB$).

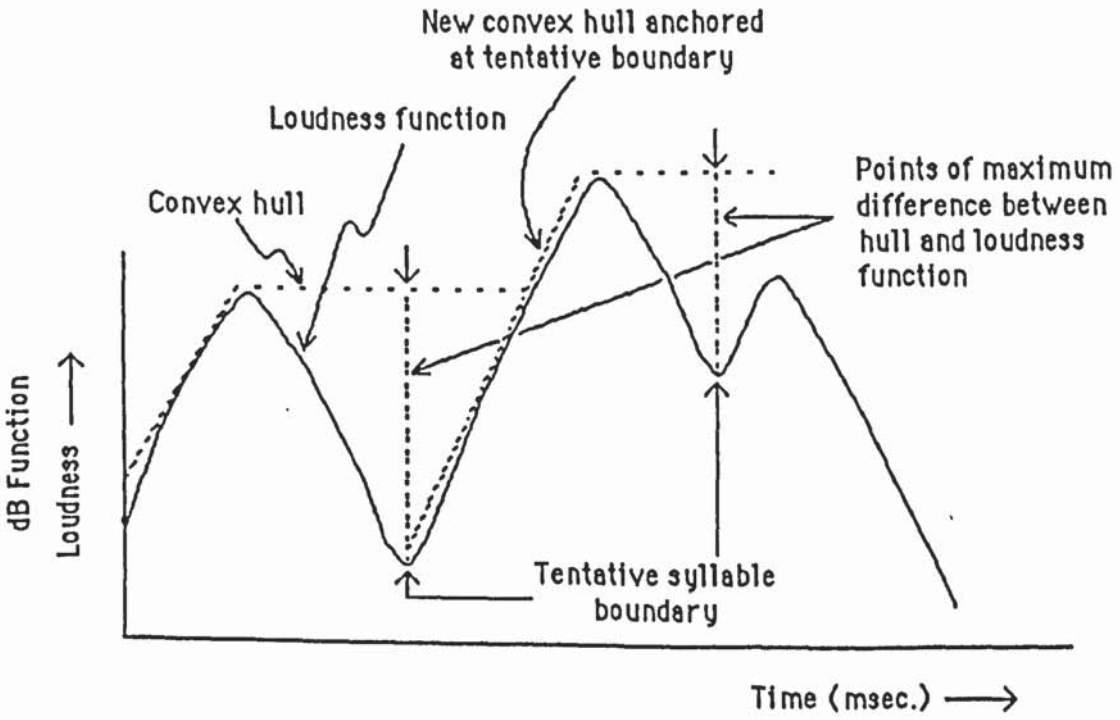


FIGURE 26

LOUDNESS FUNCTION AND THE CONVEX HULL

The tentative syllable boundary points obtained are many more than the actual number of syllables, hence all tentative syllable boundary points are subjected to some kind of a measure of significance designed to eliminate all extraneous syllable boundaries. Some of the constraints used are:

1. If two tentative syllable boundary points lie within a certain time of each other, e.g., 120 msec., then one is discarded.
2. If a tentative syllable boundary is within 100 msec. from the previous one but is not followed by another one for a period of 500 msec., then this tentative syllable boundary is regarded as a syllabic boundary.

The performance of the Mermelstein syllabic segmentation algorithm was evaluated by processing 11 sentences read by two male subjects. The overall frequency of syllable-count differences with respect to nominal syllabic count was 9.5%, consisting of 6.9% missed and 2.6% extra syllables.

6.4 Arabic Segmentation Algorithm

6.4.1 Arabic Language Characteristics and Syllable Definition.

Some sounds (phonemes) are more 'sonorous', i.e., have greater carrying power than others. The sonarity of a sound is its loudness relative to that of other sounds with the same length, stress and

pitch. The degree of vocal tract obstruction in the production of sounds determines the degree of sonarity of sounds. Therefore, vowels with the least obstruction in the vocal tract system are more sonorous than fricatives or liquids. Jespersen [47] has classified sounds according to their degree of sonarity in the following manner (beginning with the least sonorous first):

- Voiceless consonants
- Voiced consonants
- Nasals, laterals, trills
- Semi vowels
- Vowels

If we assume that the sequence of phonemes which constitute a syllable have the tendency of clustering around the most sonorous sound, then a syllable may accordingly be defined to be the distance between two minimas of sonarity. Peaks of sonarity often correspond to vowels. This definition of a syllable have sometimes failed to produce satisfactory results in terms of word syllable content for the English and many European languages, mainly because in some words consonants do not cluster around the most sonorous sound in a syllable. For example, 'spa' contains two peaks of sonarity, yet it is made up of one syllable.

Arabic on the other hand is a language that has the following main characteristics:

1. 35 distinct phonemes (6 vowels plus 29 consonants.
2. Rule based morphological system, i.e., every word can be derived from a three letter or four letter root word.
3. High correspondence of letter-to-sound rules, i.e., one to one correspondence with no exceptions except for the phoneme /?/.
4. Rule based lexical stress assignment.
5. Syllabic behaviour, i.e., six types of syllable structures can occur (CV, CVV, CVC, CVVC, CVCC and CVVCC), however only the first three are common. Maximum number of syllables forming a word cannot exceed seven, and no more than four syllables of the open type, i.e., CV or CVV can occur within a word. Phonemes groupings within a syllable obey certain rules, i.e., some consonant sequences within a syllable are not allowed, for example, /r/ cannot be followed by /l/. Closed syllable types are more common and the initial consonant cluster has to be singular, i.e., syllables must start with only one consonant. The final consonant cluster can be nil, one or two. Syllabic sounds, i.e., those occurring in syllable nuclei, are always vowels irrespective of the context.

Loudness of a sound is an auditory property and it is a function of the amplitude displacement and frequency of the pressure sound waves, i.e., the higher the amplitude displacement or frequency of sound waves, the louder the sound is. A good measure of loudness is the sound intensity (energy). For the digitized speech samples, the average magnitude function is used as a measure of loudness, as given in equation 3.

Based on the above considerations of the syllabic behaviour of the Arabic language, Arabic syllables can be defined to be the distance between two minima points of the energy contour with one maximum point lying between them, provided that this distance is not less than some defined minimum. This gives the Arabic syllabic patterns shown in Figure 27 for the different Arabic syllable types.

The high degree of correspondence between the phonetics and the acoustics of the Arabic language in support of the syllable definition given above are observed in Figures 28 and 29 for the Arabic words /kataba/ and /jataʕallam/.

Figures 28 and 29 show the pressure waveforms of the Arabic utterances /kataba/ and /jataʕallam/ and their respective energy contours. From the energy

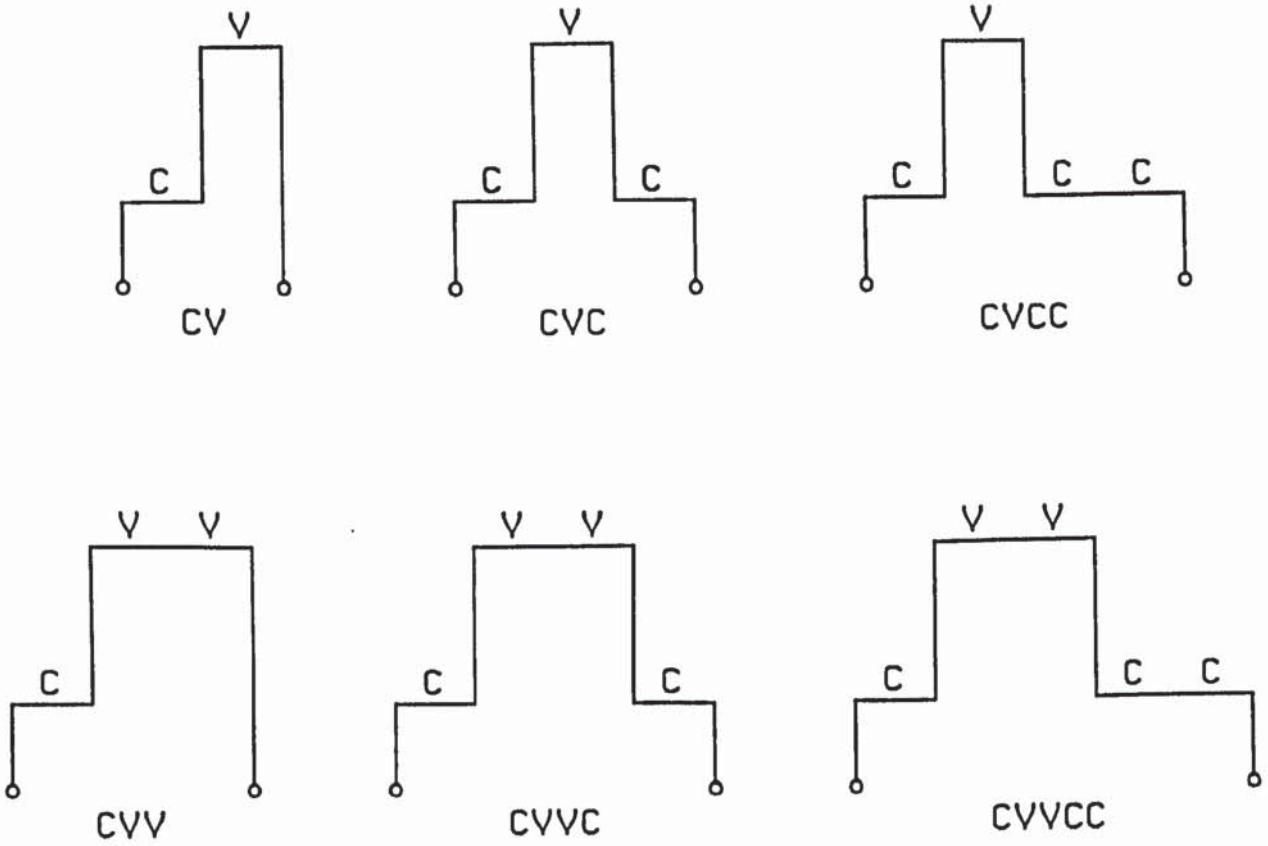


FIGURE 27

ARABIC SYLLABLE PATTERNS

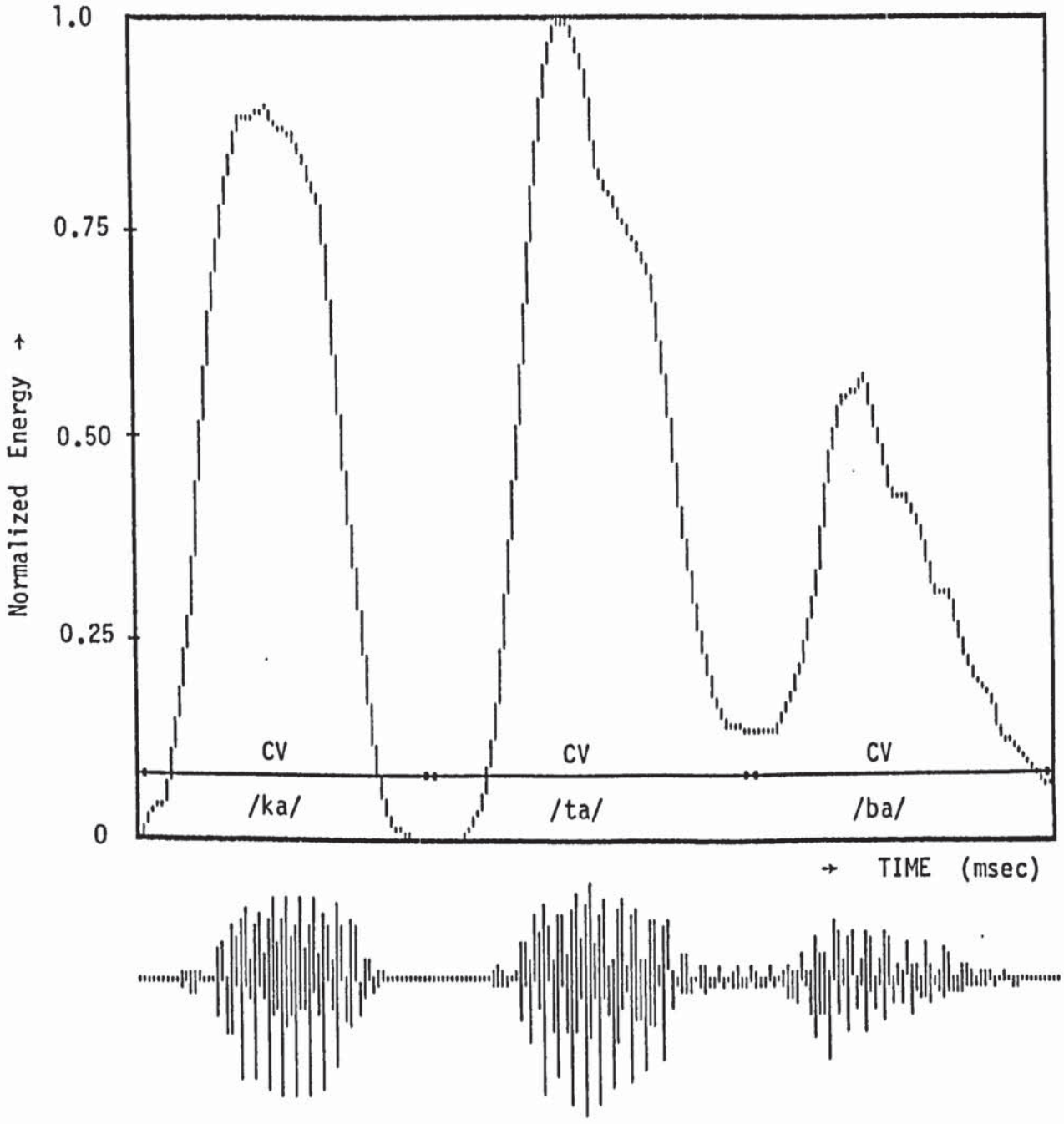


FIGURE 28

PRESSURE WAVEFORM AND ENERGY CONTOUR OF THE ARABIC WORD /kataba/.

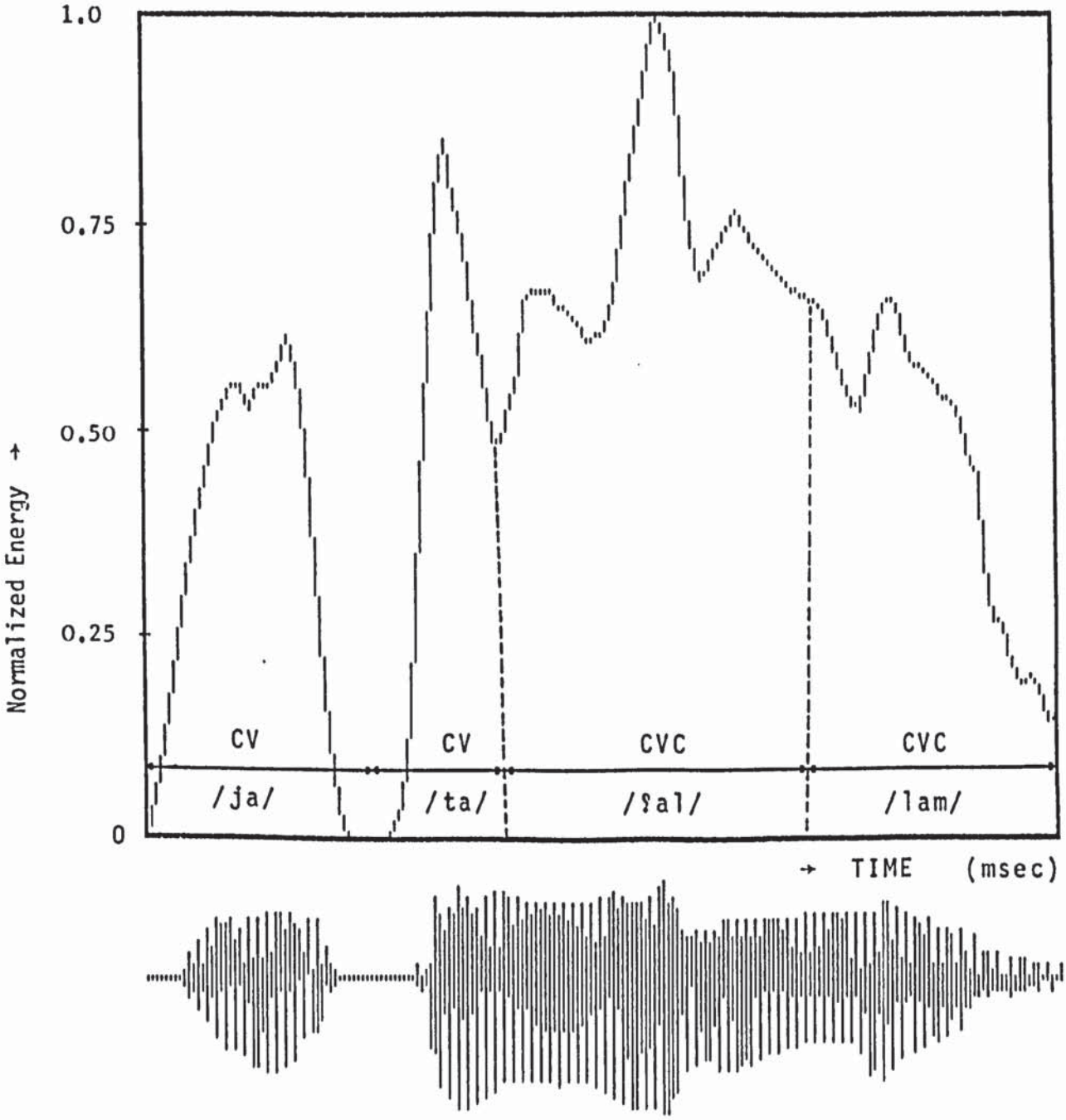


FIGURE 29

PRESSURE WAVEFORM AND ENERGY CONTOUR OF THE ARABIC WORD /jataʔallam/.

contour of the first utterance, we observe clear prominent maxima and minima points that truly reflect the syllabic structure of the word, the number of these maxima pionts equals the number of syllables present. The syllable boundaries are again very well identified by the minima points in the energy contour. The first Arabic utterance is made up of three CV syllables as indicated in Figure 28. The consonants associated with these syllables are of the stop type, and that is why we observe the deep minima points (low energy) in the energy profile. The second utterance is composed of four syllables, two of the type CV, and two of the type CVC, as indicated in Figure 29. Since the consonants associated with these syllables are now of the type semi-vowel /j/, stop /t/, pharyngeal /ʕ/, liquid /l/ and nasal /m/, respectively, the energy contour exhibits many maxima and minima points. If we use the Arabic syllable definition given earlier and pick up the most prominent maxima (highest peak among several peaks that are close together), and if these prominent maxima are far enough from each other, we can then assume that the syllable count for this utterance is equal to the number of the prominent maxima points.

6.4.2 *Aim of the Arabic Segmentation Algorithm.*

The overall aim of the Arabic syllabic segmentation algorithm is to evaluate the maxima and minima points of the energy curve profile of the speech utterance so as to decompose it into its syllabic segments.

6.4.3 *Segmentation Algorithm.*

The segmentation algorithm proceeds by performing four steps, namely, digital-filtering the speech samples, computing the utterance energy profile, algorithm for maxima/minima points-picking from the energy profile and maxima/minima points significance testing. Each step is implemented by a software procedure. A flowchart of the Arabic syllabic segmentation algorithm is shown in Figure 30. The following is a description of each subroutine procedure:

1. **Filtering**

   In order to smooth the energy contour of the speech signals, the speech samples time signal is low-pass filtered by a recursive digital filter. Since prominent energy maxima points correspond to Arabic vowel segments and since also the first formant F1 is always the strongest among the three vocally generated formants F1, F2 and F3 [48], the 3dB cutoff frequency of the digital filter is set at 1900

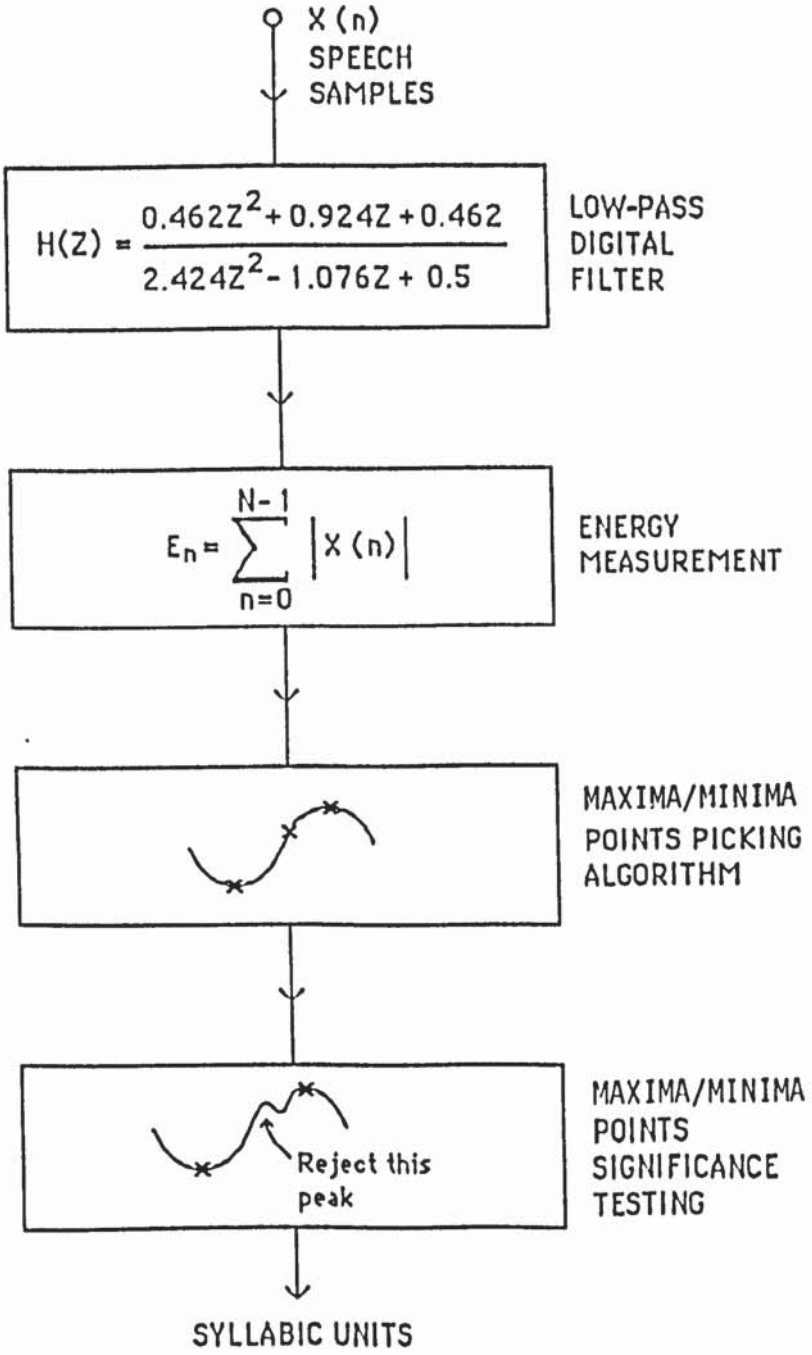


FIGURE 30

FLOWCHART OF THE ARABIC SYLLABIC SEGMENTATION ALGORITHM.

Hz. This emphasizes the first formant relative to the other higher formants. The Z transform of the digital filter is given in equation 9. It is arrived at by applying the bilinear transform method [49] which transform a low-pass analog filter (Butterworth type of order 2) into a recursive digital low-pass filter.

$$H(z) = \frac{0.462Z^2 + 0.924Z + 0.462}{2.424Z^2 - 1.076Z + 0.500} \qquad (9)$$

The frequency response of this low-pass digital filter is shown in Figure 31.

A canonical-form realization of this low-pass digital filter is shown in Figure 32. Once the speech samples are filtered, the speech utterance amplitude is normalized so that the largest filtered speech sample amplitude has a value of one. Normalization of the filtered speech samples reduces the speech sample amplitude variability resulting from different utterances without altering the relative changes in amplitude within a given utterance. Let the filtered speech utterance be represented by the train of speech samples x(i), where, $0 \leqslant i \leqslant k$. Also, let max|x(i)|,

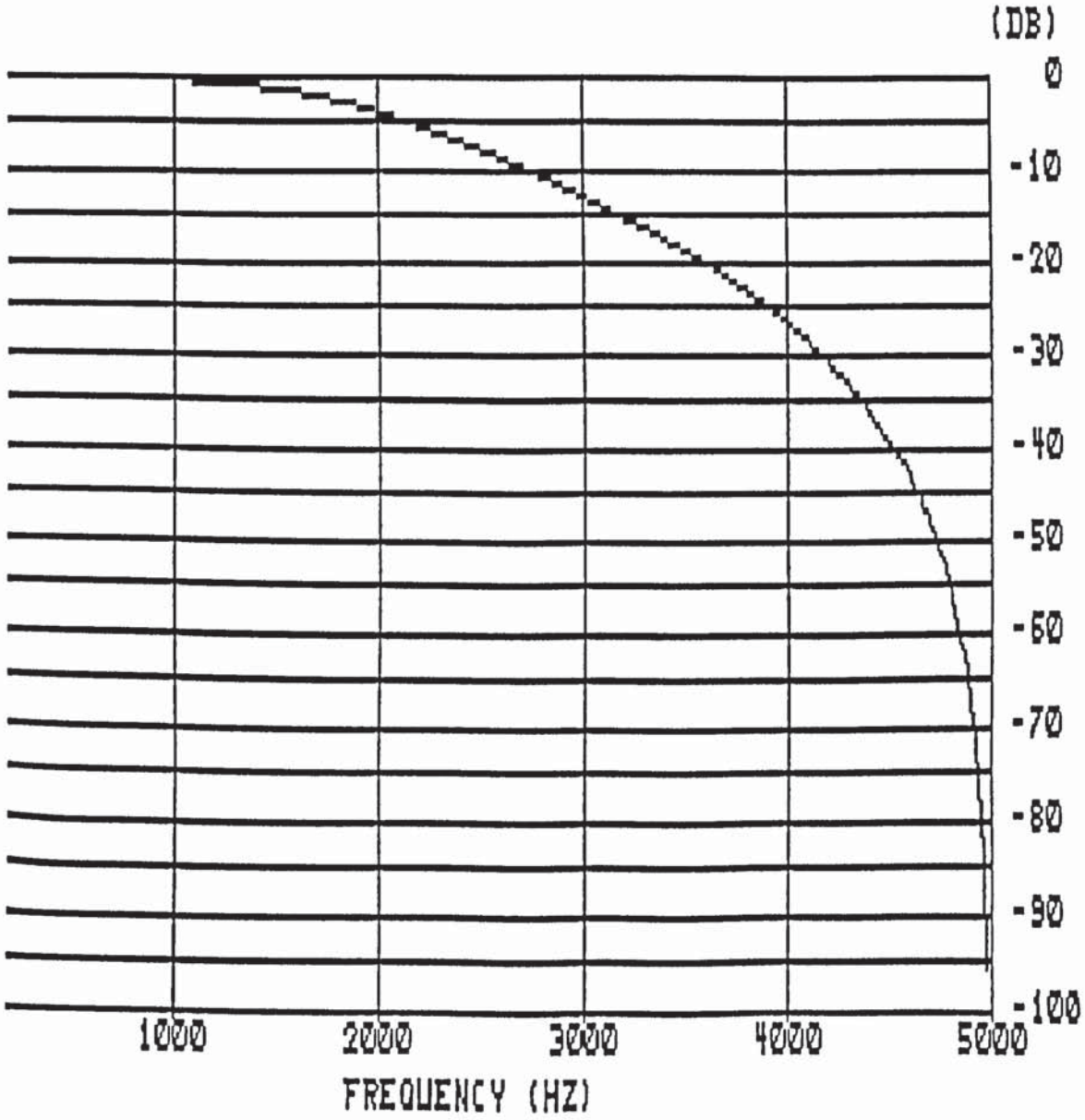
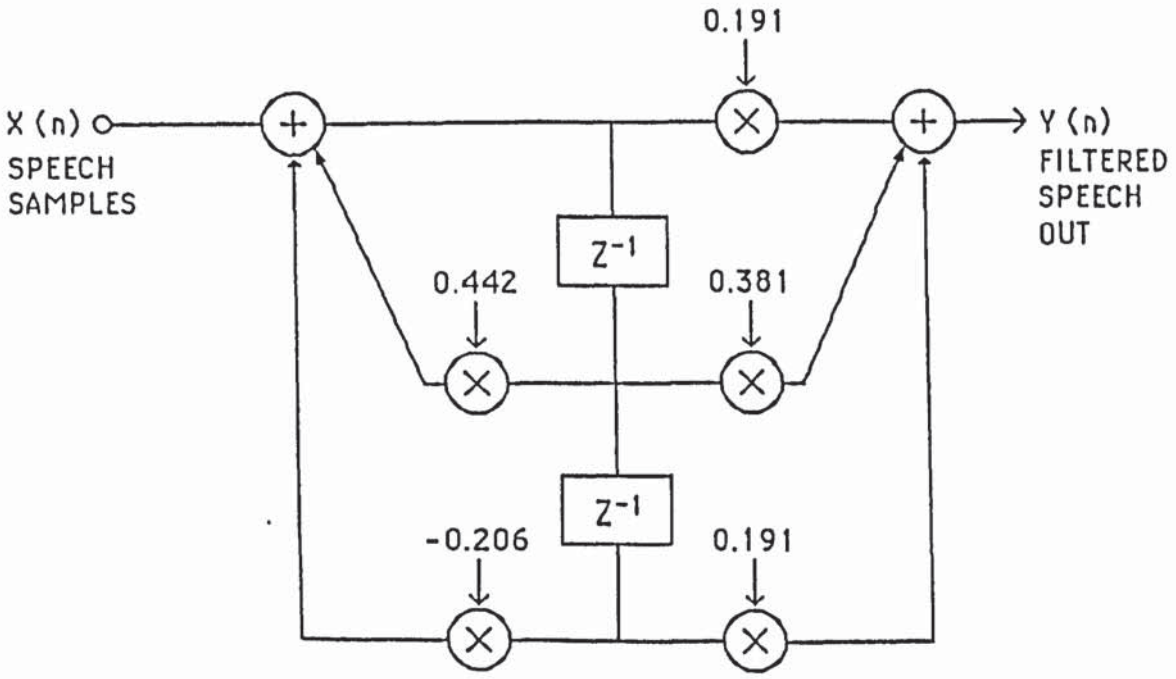


FIGURE 31

FREQUENCY RESPONSE OF LOW-PASS DIGITAL FILTER.



$$H(Z) = \frac{0.191Z^{-2} + 0.381Z^{-1} + 0.191}{1 - 0.442Z^{-1} + 0.206Z^{-2}}$$

FIGURE 32

REALIZATION OF THE LOW-PASS DIGITAL FILTER

$0 \leq i \leq k$ be the largest filtered speech sample. Then the normalized filtered speech samples are simply $x(i)/\max_{0 \leq i \leq k} |x(i)|$ for all, $0 \leq i \leq k$. The normalized filtered speech samples are stored in array FDATA.

2. **Speech Energy Profile**

The usual form of energy definition of a discrete-time real signal x(n) is as given in equation 2. This form of energy definition is very sensitive to large signal levels, thereby emphasizing large sample-to-sample variations. To alleviate this problem and to accommodate the desire for faster computation, the magnitude function as given in equation 3 is used. This form of energy definition has the advantage of producing a smooth energy function. This technique for energy measurements was also used by Rabiner [24].

The energy $\hat{E}(k)$ of the speech signals is calculated in frames of 256 speech samples in length, windowed by a Hamming window w(n) of size 256, and overlapped by a quarter frame length according to the following routine put in PASCAL form:

```
begin
 l=0; (integer)
 k=0; (integer)
  Repeat
```

$$\hat{E}(k) = \sum_{n=0}^{N-1} |x(l+n)\ w(n)|$$

```
   l=l+N/4;
   k=k+1;
  until l+N> end point of speech;
End.
```

where,

x(n) - speech sample at unit time n,

w(n) - [0.54 - 0.46 cos $(2\pi n/N-1)$]

$0 \leqslant n \leqslant N-1$, and

N - Hamming Window size - 256

The result from the application of this routine is the formation of the array SUM which, when plotted, will display the energy contour of the utterance.

3. **Maxima/Minima Points Picking Algorithm**

This procedure is responsible for picking all the maxima and minima points of the energy contour stored in array SUM. The result is the formation of two arrays EMAX and EMIN, containing the locations of all the maxima and minima points, respectively. A maxima energy point is determined once the slope of energy

curve changes sign from positive to negative and vice-versa for the minima energy point. The following two loops are used to collect all maxima and minima points of the energy contour, these loops are repeated until all points of the array SUM are exhausted.

```
Repeat
   Repeat
      if SUM[i+1]-SUM[i] <0  Then EMAX[k]=i;
      i=i+1;
   Until EMAX[k]≠0 OR i=end of array SUM
   (once a maxima point is obtained we look
    for a minima point)

   Repeat
      if SUM[i+1]-SUM[i] >0  Then EMIN[k]=i;
      i=i+1;
   Until EMIN[k]≠0 OR i=end of array SUM
  k=k+1;
Until pointer i reached end of array SUM
```

The flowchart of the maxima and minima points picking algorithm is shown in Figure 33. Generally, there will be many more maxima than the number of syllables. This can be attributed to the nature of speech signals in terms of its complexity since, besides phonology, it carries other information such as prosody and physiology. Maxima and minima of the energy contour are potential syllable nuclei and syllable boundaries, respectively.

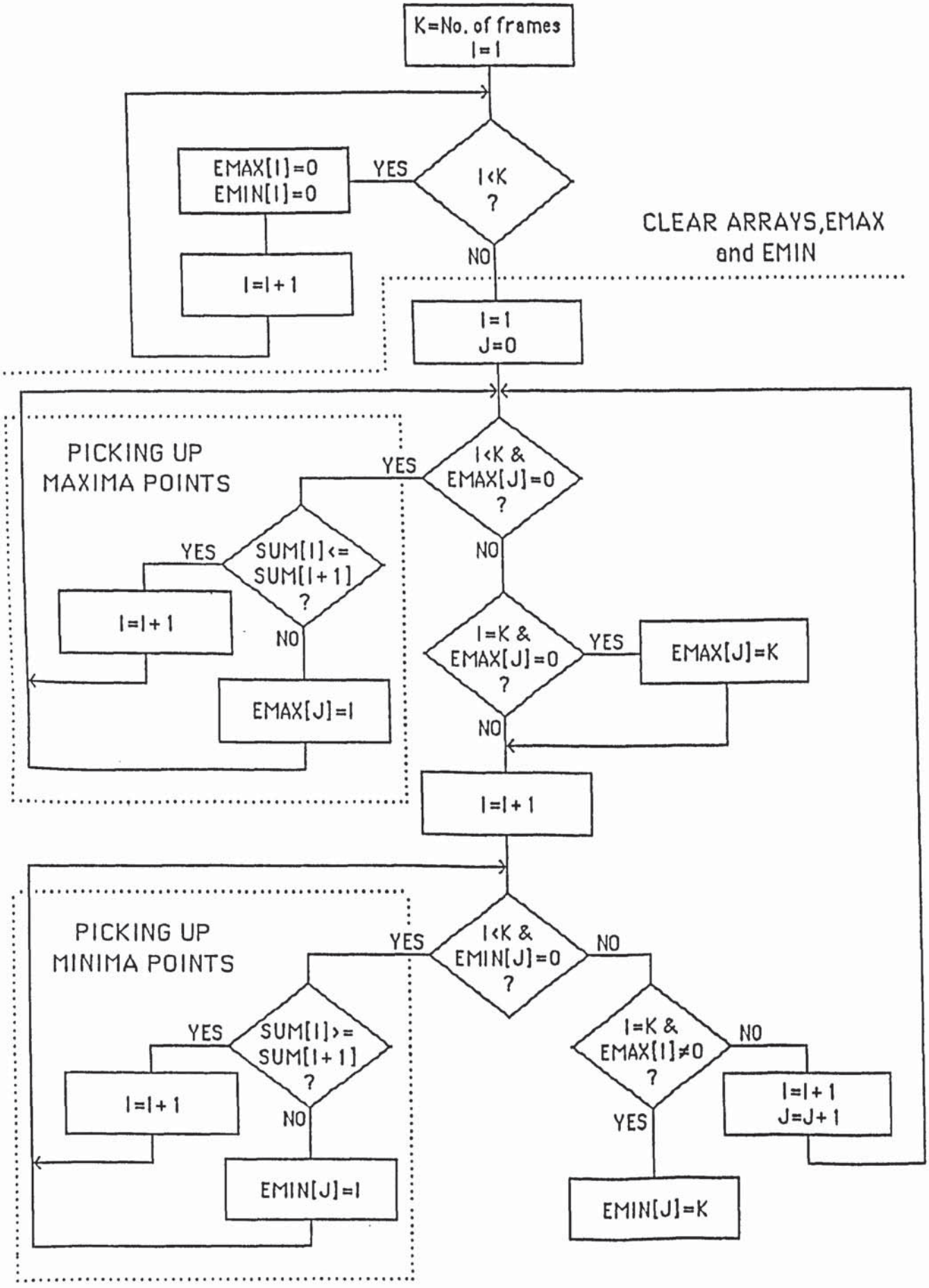


FIGURE 33

FLOWCHART OF MAXIMA/MINIMA POINTS PICKING ALGORITHM.

4. Maxima/Minima Points Significance Testing

In order to arrive at potential syllabic nuclei and boundaries, some form of significance testing has to be applied to the collected maxima and minima points. This is equivalent to smoothing the energy curve by eliminating false and misleading maxima and minima points created by the complex nature of the speech signal.

In order for the syllabic segmentation process to be effective, the evaluation measures have to be context dependent. Basically, the collected maxima and minima points of the energy contour are subjected to three kinds of measures. Each measure can be looked at as one level of a smoothing process. The measures are:

a. In a set of up to three consecutive maxima points within 19.2 msec., the highest energy maxima point is selected and the rest is rejected if the maxima points and their associated minima points form a positive energy curve slope (energy increasing trend). Similarly, in a set of up to three consecutive minima points, the lowest energy minima point is

selected and the rest are rejected if the minima points and their associated maxima points form a negative energy curve slope (energy decreasing slope). In this way, maxima and minima points forming a staircase shape are rejected.

b. Two threshold constants, dx and dy are calculated. dx=4% of total utterance duration and dy=10% of the highest maximum energy point value. These threshold consonants are used to evaluate the significance of each maximum point and its associated minima points, (i.e., two minima points, one falling on the left-hand side of the maximum point and the other on the right-hand side). The evaluation process proceeds as follows:

Let EMAX[i] denote the maxima point under consideration, and EMIN[k] and EMIN[k-1] denote the minima points on the right and left hand sides of maxima point EMAX[i]. The maxima point EMAX[i] is rejected if (EMIN[k]-EMAX[i] < dx AND SUM[EMAX[i]-SUM[EMIN[k]] < dy). Once the test is true, (i.e., EMAX[i] is rejected), then the highest energy point of the two minima points is also rejected. The same process is repeated

for the left hand side minimum point (EMIN (k-1)) if the test on the right hand minimum point has failed, i,e., energy points EMAX(i) and EMIN(k) are not within the thresholds dx and dy.

Each maxima point is subjected to the same evaluation until all maxima points have been evaluated. Figure 34 shows the flow-chart implementation of this process.

c. Minima (dip) points (potential syllabic boundaries) of energy values above 70% of the highest energy point are rejected. Similarly, maxima (peak) points (potential syllabic nuclei) of energy values less than 15% of the highest energy point are rejected.

6.5 Test Speech Vocabulary

The speech utterances used to test the Arabic syllabic segmentation algorithm consisted of 50 monosyllabic and 82 multisyllabic words spoken by three males and one female, referred to as informants. Each informant made on the average five repetitions per word, thus a total of more than 2640 Arabic words were used to test the Arabic syllabic segmentation algorithm. The uttered words are digitized via the developed PC-based Arabic speech

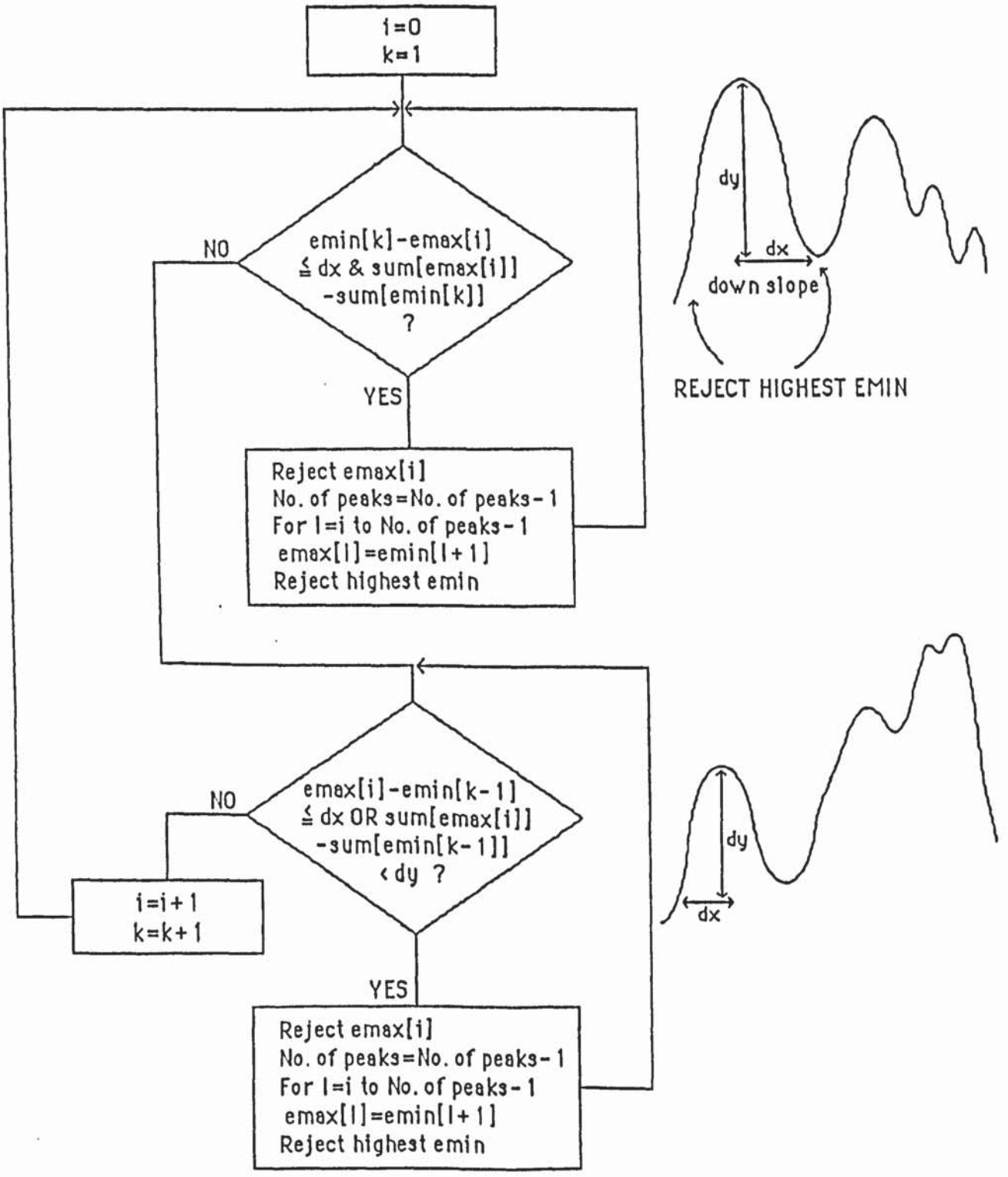


FIGURE 34

FLOWCHART OF ENERGY CURVE SMOOTHING BY APPLICATION OF THRESHOLD CONSTANTS dx AND dy.

processing system explained in Chapter 4, and are stored in speech files. Recordings of uttered Arabic words were not made in one session, hence the effect of time variations is included.

6.6 Software System Implementation

The Arabic syllabic segmentation algorithm was realized by the set of PASCAL routines given in Appendix D.

6.7 Syllabic Segmentation Example

As an example of the segmentation process for Arabic utterances into their syllabic units, the word /Sajja:d/ spoken by the author is used. This word consists of two syllables of the types, CVC and CVVC. The initial consonant of the first syllable is the Arabic emphatic sound /S/, the final consonant of the first syllable and the initial consonant of the second syllable are the same, i.e., the geminated semi-vowel /j/. The final consonant in the second syllable /d/ is of the voiced stop type.

Once the speech file containing the word to be segmented is read, end point detection is performed manually. This is done by displaying the speech pressure waveform on the PC screen, and then through cursor movements, end points are readily detected. The Arabic syllabic segmentation process is initiated by first low-pass digital filtering of the speech samples followed by the energy profile computation.

The maxima/minima picking algorithm is then invoked, the result being a set of maxima and minima points. Figure 35 shows the energy contour with maxima and minima points marked by up ↑ and down ↓ pointing arrows, respectively.

The following are the results obtained after applying the above procedures to determine the appropriate syllabic units:

* Number of frames analyzed - 88.
* Number of speech samples points per analysis frame - 256.
* Maxima and minima points detected and stored in EMAX and EMIN arrays respectively are:
  - EMAX 1 6 23 53 61 84
  - EMIN 0 2 8 37 56 83 88
* dx - 4, dy - 3.8 dB.
* Maxima and minima points left after smoothing by threshold values dx and dy are:
  - EMAX 23 61
  - EMIN 2 37 88
* All maxima points have energy values that are greater than 15% of the highest energy point. Similarly, all minima points have energy values that are less than 70% of highest energy point. So the application of the last constraint is not required. Hence, the number of syllables arrived

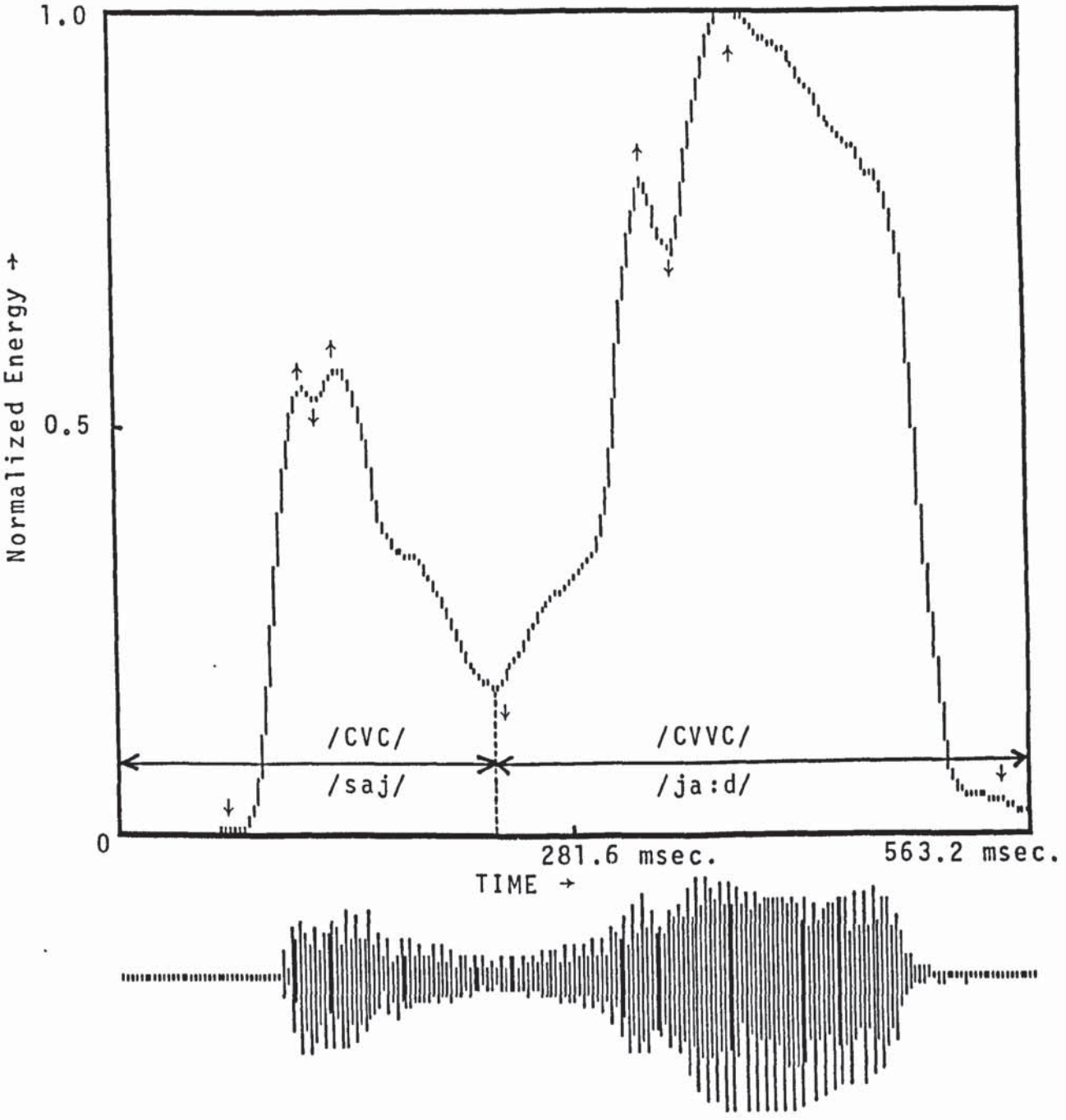


FIGURE 35

MAXIMA AND MINIMA POINTS OF THE ENERGY CONTOUR OF THE ARABIC UTTERANCE /Sajja:d/.

at are two, at maxima points 23 and 61. The Arabic syllabic unit durations are 224 mseo. and 326.4 mseo. for the first and second syllabic units, respectively.

6.8 *Syllabic Segmentation Results*

The performance of the Arabic syllabic segmentation algorithm described above was evaluated by processing the 50 monosyllabic and 82 multisyllabic words, spoken by the informants. Words chosen were those having the variety of syllable types and syllable structures of the Arabic language outlined in Section 2.1. The testing process of the Arabic syllabic segmentation algorithm proceeded in the following manner:

1. First the digitized Arabic words were used to tune-in the threshold constraints values. The following threshold constraints values were found to produce satisfactory Arabic syllabic counts.
   - dx = 4% of total utterance duration.
   - dy = 10% of the highest energy point value.
   - The upper limit energy value of any minima point to be considered as potential syllable boundary is equal to 70% of the highest energy point.
   - The lower limit energy value of any maxima point to be considered as potential

syllable nucleus is equal to 15% of the highest energy point.

The dx threshold value was found to be more critical than the others in obtaining correct syllabic count because it relates to syllable duration, i.e., low dx values result in extra syllabic counts, i.e., syllable fragments are generated. On the other hand, the other threshold values were found to be less critical in affecting syllabic counts. For example, the dy value can vary between 8-12% of the maximum energy point. Similarly, the upper and lower energy limits of the minima and maxima points may vary between 65-70% of maximum energy point, and 10-15% of maximum energy point, respectively.

2. Secondly, the performance of the Arabic syllabic segmentation algorithm was tested. The accuracy of Arabic syllabic counting using the 50 monosyllabic words was 96%, using 39 di-syllabic words 84%, and using 43 tri-syllabic words 93%. These results are shown in the form of a confusion matrix displayed in Table 9.

Extra syllable counts are caused by syllabic structures containing the consonant /r/, because the phoneme /r/ possesses distinct

formant structures which are interrupted by a short gap. This gap appears in the middle of the resonance as shown in Figure 36 for the word /magrib/. This gap represents a cut-off of energy as the tip of the tongue taps against the hard palate. The energy contour will therefore have a minima point which is picked up by the maxima/minima points picking algorithm.

The overall accuracy of the segmentation process is 92%. The syllabic boundaries located by the algorithm are simply related to the actual syllabic boundaries. The algorithm has also proved to be a very important tool in phonological analysis as stressed syllables can easily be identified by examining syllable durations and the energy of maxima points.

6.9 *Conclusion*

An automatic segmentation algorithm suitable for partitioning Arabic speech into Arabic syllabic units has been developed. The algorithm was tested with monosyllabic and multisyllabic Arabic words and its overall performance was satisfactory. The main parameter affecting the reliability of the derived syllabic count are the threshold values dx and dy, which are context sensitive. The segmentation algorithm is computationally simple to realize and

Actual Syllable \ Syllabic Count	1	2	3	4	Count Accuracy
1	48	2	—	—	96%
2	1	33	5	—	84%
3	—	2	40	1	93%

Overall Accuracy = 92%

Table 9

Syllabic Count Confusion Matrix.

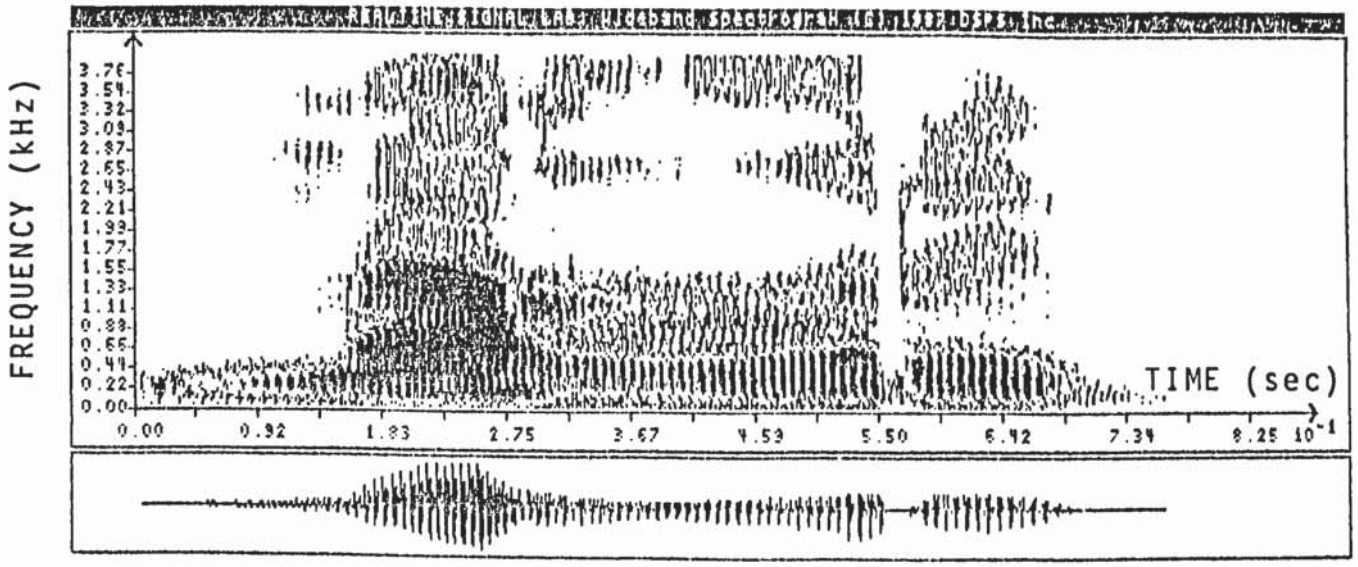


FIGURE 36

SPECTROGRAM OF THE ARABIC UTTERANCE /magrib/.

execute, as will be demonstrated later. It has many important applications in Arabic speech processing, especially in speech recognition.

CHAPTER 7

Chapter 7

ARABIC SPEECH RECOGNITION APPLICATIONS

7.1 Introduction

Automatic speech recognition is the process of transforming continuously varying acoustic signals into discrete linguistically defined sound units. The size of these units can be as minimal as possible, i.e., phonemes or as large as possible, i.e., words or phrases of sentences. Normally sound units such as phonemes, syllables and words have been used. For a speech recognition system to work with acceptable error rates, one has, to a certain extent, to imitate the human listener system process which uses phonological, syntactic and semantic information to understand the message being transmitted from the sender. The redundancy of the language allows the speaker to be imprecise with his articulations, and hence cause what is called "speech variability problem". This is not the only problem, other problems do exist such as speaker independence, noise from auditory environment and vocabulary size. The speech variability problem is demonstrated in the spectrograms shown in Figures 37 and 38 for the spoken utterance /jatma.a/ "he walked", uttered by the author twice. It is quite evident that the

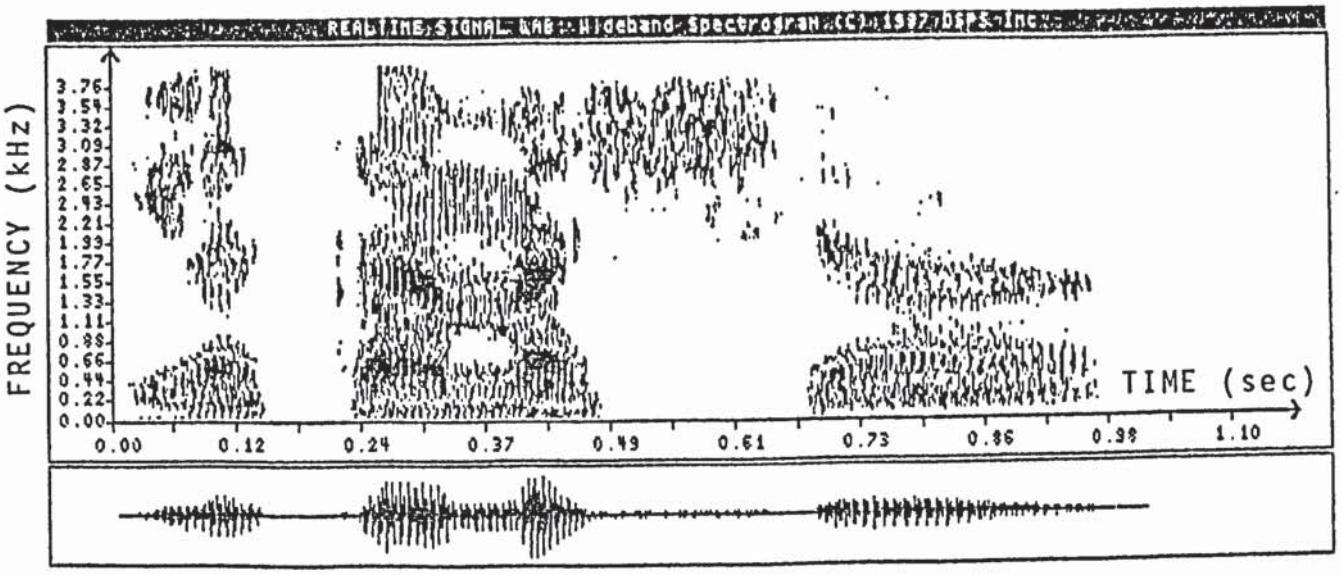


FIGURE 37

SPECTROGRAM OF THE UTTERANCE "HE WALKED".

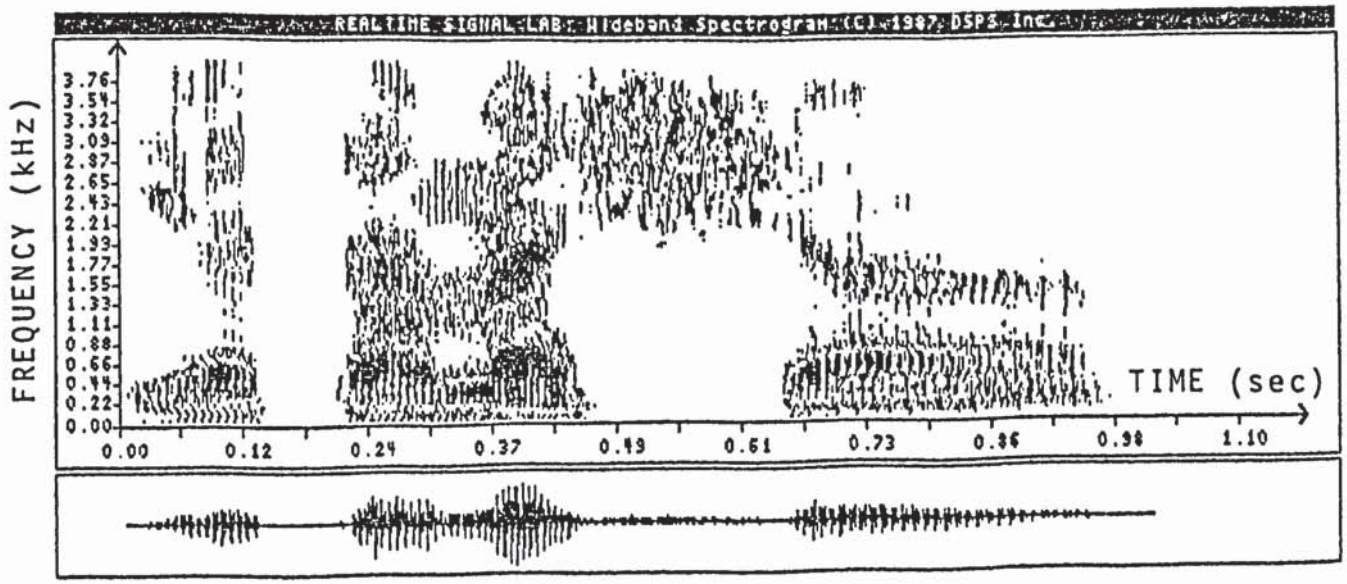


FIGURE 38

SPECTROGRAM OF THE UTTERANCE "HE WALKED".

spectral features (formant frequencies, energies and fundamental frequencies) are different. The characteristic requirement of a realistic automatic word recognition system was mentioned in Section 3.3.

Many commercial and experimental automatic speech recognition systems exist today. These systems take the form of isolated, connected and continuous word recognition systems. The developed speaker independent Arabic automatic syllabic segmentation algorithm is used in the implementation of three important aspects of speech recognition applications, namely, the automatic Arabic vowel recognition system, the isolated Arabic word recognition system and an acoustic-phonetic (syllabic) Arabic module.

7.2 Automatic Arabic Vowel Recognition System

Automatic detection of Arabic vowel regions is facilitated through the application of the Arabic syllabic segmentation algorithm described earlier (Chapter 6). Once syllable nuclei (vowels) are detected, appropriate short time speech processing techniques are applied and a set of features are extracted followed by discriminant analysis techniques, where an unknown class represented by a set of features (vector space) is classified into one of the six different Arabic vowel classes. The strategy followed is that of the classical pattern recognition techniques as shown in Figure 39.

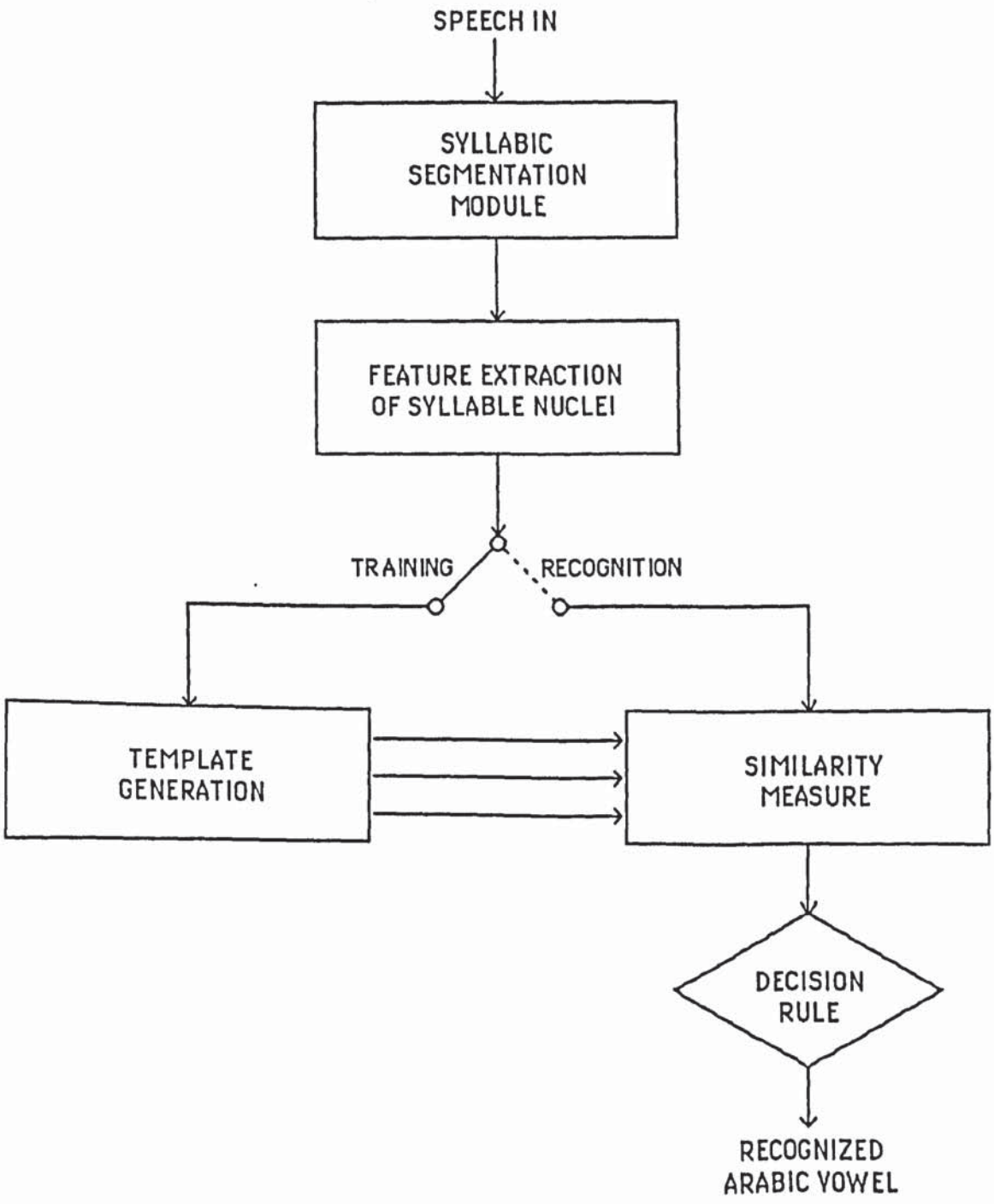


FIGURE 39

ARABIC VOWEL RECOGNITION STRATEGY.

Initially the classifier is trained with the set of features extracted during the training phase, belonging to the known Arabic vowel classes, until a set of reference or library vectors are compiled called "templates".

In the recognition mode, an unknown test vector is compared against each reference template with an appropriate similarity measure (distance measure). Choice is then made to that reference vector giving rise to maximum similarity or smallest distance found. Similarity measures are discussed in Section 7.2.2.

Our objective is thus to classify an unknown test vector representing the unknown Arabic vowel into one of the six known Arabic vowels (6 classes), assuming multivariate gaussian distribution. In order to evaluate the performance of the recognition process two problems need to be looked at, namely feature selection and evaluation, and choice of decision rule used.

7.2.1 Feature Selection and Evaluation.

The key processing step in pattern recognition systems is the feature extractor stage. Characteristic speech parameters are assumed to be stationary in the short period speech segment, typically in the order of 10 msec. in duration. Various mathematical techniques can be applied to

these 10 msec. frames, resulting in different sets of features representing the characteristic speech parameters of that frame. Techniques such as Linear Predictive Coding (LPC), Fast Fourier Transforms (FFT) and autocorrelation analysis are the most common found in literature [50]. Typically one technique is favored against another in terms of its simplicity, relative ease of computation and its parameters interpretations. The performance of one parametric representation against another was evaluated in terms of giving maximum separation of the different Arabic vowel classes. The following parametric representations were used in the investigation:

1. 25 FFT coefficients.
2. 20 Cepstrum coefficients.
3. 14 Normalized autocorrelation coefficients.
4. 14 LPC reflection coefficients.
5. 14 LPC prediction coefficients.
6. F1, F2, F3, formants estimates normalized with respect to F4.

Extraction of these features is facilitated through the use of the developed software routines that run on the Arabic PC-based speech processing system described in Appendix C.

In discriminant analysis of statistics, within-class, W, and between-class, B, covariance

matrices are used to formulate a criterion of class separation [51]. Let f_i represent any feature vector from class i. Also let $\bar{f}_i$ denote the mean feature vector for the i^{th} class, i.e., $\bar{f}_i = \langle f_i \rangle$, where $\langle\rangle$ denote expectations. Then the within-class (intraclass) covariance matrix for class i is,

$$W_i = \langle (f_i - \bar{f}_i) \ (f_i - \bar{f}_i)^T \rangle \tag{10}$$

where, T denotes transpose.

Now, the pooled within-class (intraclass) covariance matrix for all classes is simply the average of the individual covariance matrices of the classes, i.e., $W = \langle W_i \rangle$. If we represent each class by its mean feature vector, for example $\bar{f}_i$ to represent class i, then the between-class (interclass), B, covariance matrix is,

$$B = \langle (\bar{f}_i - \bar{f}) \ (\bar{f}_i - \bar{f})^T \rangle \tag{11}$$

where $\bar{f}$ is the mean of f_i overall classes. Hence, W measures the scatter within the classes and B measures the scatter between the classes. To see how the two covariance matrices, i.e., W and B, are used to formulate a criterion of class separation, let us examine the case where only two types of classes are

involved. Suppose we make a sufficient number of measurements using two types of feature vectors z_1 and z_2, and we find the distributions to be similar to the sketches shown in Figures 40 and 41. It is then obvious from the Figures 40 and 41 that the feature vector z_1 does a better job at separating the two classes because there is no overlapping between the distributions, although the difference between the two means of each class is less than that using the feature vector z_2. Overlapping between the two distributions depends on the separation of the means of the classes and the width of the class distribution, i.e., B and W. Accordingly, this means that greater separability of classes, i.e., less overlapping in distribution, is achieved if the covariance matrix B (Between) is made as large as possible, and the covariance matrix W is made as small as possible. Put in a numeric form, we are then looking for a feature that maximizes the ratio of B/W. Typical criteria of class separability quoted in the literature [52] is $|B|/|W|$. In order to compute and pictorially represent the separation of the Arabic vowel system using the different sets of features described earlier, a canonical analysis procedure is performed [53].

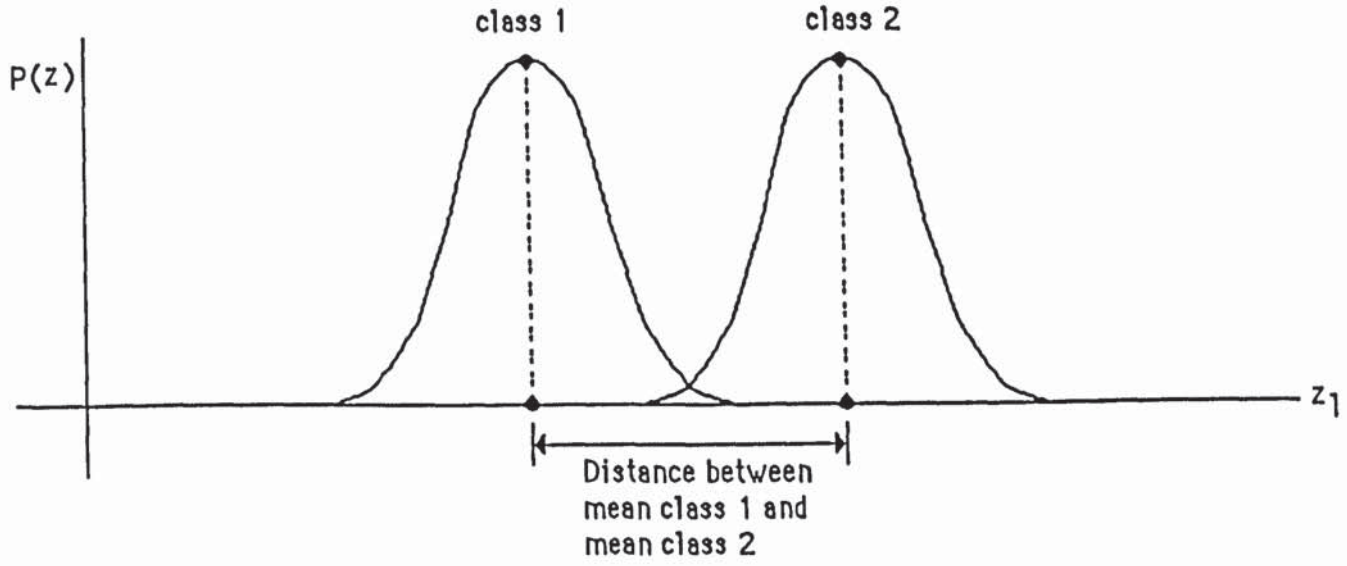


FIGURE 40

DISTRIBUTION OF CLASSES 1 AND 2
USING FEATURE VECTOR z_1.

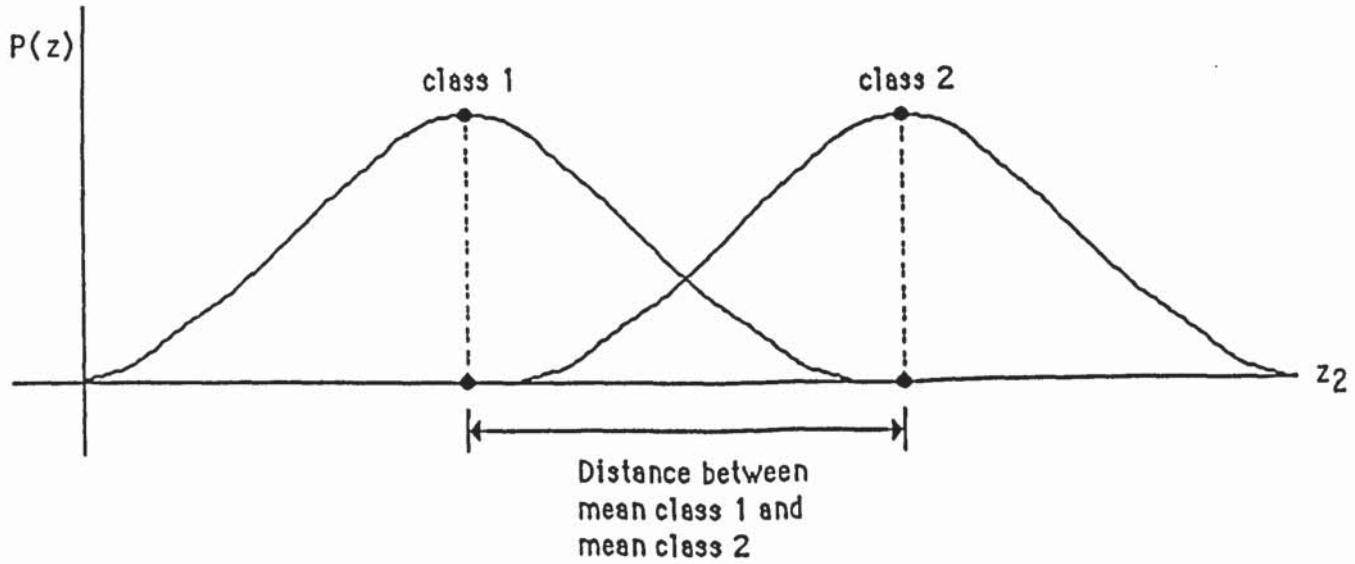


FIGURE 41

DISTRIBUTION OF CLASSES 1 AND 2
USING FEATURE VECTOR z_2.

A program called "Candisc", part of a statistical analysis system package (SAS) running on the IBM mainframe, is utilized to generate the scatter diagrams of the Arabic vowel system using these different sets of features. Canonical analysis is concerned with the dimensional reduction of the feature space. The mathematical formulation behind this technique is described in the literature [54]. The output from this analytical procedure are two canonical variables, CAN1 and CAN2 which summarize the between-class variations. CAN1 and CAN2 are linear combinations of the variables that constitute a feature vector. A two-dimensional plot showing the separation of the different classes is achieved using CAN1 and CAN2 for the axis.

7.2.2 Decision Rule.

Classification of an unknown test vector into one known class is done through the application of an appropriate similarity measure made against all reference templates. Decision is made in favor of one or more particular reference templates depending on the highest similarity measure obtained. The best rule is that of Baysian statistics which states that choice is made in favour of one class against another if the conditional probability of that class, given some measured set of features, is greater than the probability of the other class given the same set of

measured features [52]. Let C_i be one vowel class type and C_j be the other vowel class type, and let vector f be the measured set of features, then choice is made in favour of vowel class C_i if,

$$P\{C_i|f\} > P\{C_j|f\} \tag{12}$$

or

$$P\{C_i|f\} \; / \; P\{C_j|f\} > 1 \tag{13}$$

It is very difficult to estimate the conditional probability of the different vowel classes given the set of measurement features as it stands, i.e., $P\{C_i|f\}$. Fortunately, using Bayes formula we can obtain another kind of conditional probability which is easier to estimate. For example the conditional probability $P\{f|C_i\}$, i.e., the probability of getting one particular set of features (measurements), f, given that the measurement comes from vowel class C_i, is something that can be estimated simply by taking a reasonable number of measurements from vowel class C_i. $P\{C_i|f\}$ and $P\{f|C_i\}$ are related by Bayes theorem [55]. Using Bayes theorem we get,

$$P\{C_i|f\} = \frac{P\{f|C_i\}\ P\{C_i\}}{\sum_{all\ k} P\{f|C_k\}\ P\{C_k\}} \qquad (14)$$

$$P\{C_j|f\} = \frac{P\{f|C_j\}\ P\{C_j\}}{\sum_{all\ k} P\{f|C_k\}\ P\{C_k\}} \qquad (15)$$

where, $P\{C_i\}$ denotes the probability of vowel class C_i, i.e., the proportion of vowel class C_i in the population. If we assume all vowel classes are equally likely, then $P\{C_i\} = P\{C_j\} = \ldots$ *etc* = 1/6 for the Arabic vowel system. Hence, our classification rule of equation 14 transforms to,

assign to vowel class C_i if,

$$\frac{P\{f|C_i\}\ P\{C_i\}}{\sum_{all\ k} P\{f|C_k\}\ P\{C_k\}} > \frac{P\{f|C_j\}\ P\{C_j\}}{\sum_{all\ k} P\{f|C_k\}\ P\{C_k\}} \qquad (16)$$

for all $j \neq i$

cancelling the bottom line of both sides of the inequality, equation 16 reduces to,

$$P\{f|C_i\}\ P\{C_i\} > P\{f|C_j\}\ P\{C_j\} \qquad (17)$$

for all $j \neq i$

Although $P\{C_i\}$ is relatively easy to estimate, $P\{f|C_i\}$ is still not that easy to estimate. However, if the features constituting the feature vector, f, have a multivariate normal distribution, $P\{f|C_i\}$ may be expressed as:

$$P\{f|C_i\} = (2\pi)^{-n/2} \mid R_i^{-1/2} \mid \exp\{-\frac{1}{2}(f-\bar{f}_i) \quad (18)$$
$$R_i^{-1}(f-\bar{f}_i)^T\}$$

where $\bar{f}_i$ equals mean feature vector of class C_i, R_i is the covariance matrix of class C_i, and n is the feature space. Therefore, to estimate $P\{f|C_i\}$ we need simply to estimate $\bar{f}_i$ and R_i. If we replace $P\{f|C_i\}$ in equation 17 with that of equation 18, we get,
assign to vowel class C_i if,

$$(2\pi)^{-n/2}|R_i^{-1/2}|\exp\{-\frac{1}{2}(f-\bar{f}_i)\ R_i^{-1}(f-\bar{f}_i)^T\}\ P\{C_i\} > \quad (19)$$
$$(2\pi)^{-n/2}|R_j^{-1/2}|\exp\{-\frac{1}{2}(f-\bar{f}_j)\ R_j^{-1}(f-\bar{f}_j)^T\}\ P\{C_j\}$$

for all $j \neq i$

Taking the natural log and cancelling all common terms of equation 19, it will reduce to,
assign to vowel class C_i if,

$$-\ln |R_i| - (f-\bar{f}_i)\, R_i^{-1} (f-\bar{f}_i)^T + \ln P\{C_i\} \;>\; -\ln |R_j| - (f-\bar{f}_j)\, R_j^{-1} (f-\bar{f}_j)^T + \ln P\{C_j\} \qquad (20)$$

for all $j \neq i$

If we assume that all vowel classes are equally likely and multiply both sides of the inequality by -1, then we will get,

assign to vowel class C_i if,

$$+\ln |R_i| + (f-\bar{f}_i)\, R_i^{-1} (f-\bar{f}_i)^T \;<\; +\ln |R_j| + (f-\bar{f}_j)\, R_j^{-1} (f-\bar{f}_j)^T \qquad (21)$$

for all $j \neq i$

The term $\ln |R_i| + (f-\bar{f}_i)\, R_i^{-1} (f-\bar{f}_i)^T$ is referred to as $D_i(f)$, the discriminant function. It cannot be thought of as a distance measure because of the term $\ln |R_i|$. However, several assumptions can be made which will result in the following distance measure [56].

1. *Mahalanobis Distance:* The assumption here is that the correlation between the features of the measurement vector, f, are the same within each vowel class, and that the correlations themselves are independent of the class. Then

all classes will have the same covariance matrix, i.e., $R_i = R_j = \ldots$ etc $= R$.

Then the function $D_i(f)$ reduces to,

$$D_i(f) = (f-\bar{f}_i)\, R^{-1}(f-\bar{f}_i)^T \qquad (22)$$

which is referred to as the Mahalanobis distance measure.

2. *Weighted Euclidean Distance:* The assumption here is that the features of the measurement vector, f, are uncorrelated and independent of the class, i.e., all classes will have the same covariance matrix. Hence $R_i = R_j = W =$ diag $(\sigma_1^2, \sigma_2^2, \ldots, \ldots)$, where σ_i^2 is equal feature variance of W at (i, i). Then $D_i(f)$ reduces to,

$$D_i(f) = (f-\bar{f}_i)\, W^{-1}(f-\bar{f}_i)^{-T}, \qquad (23)$$

which is referred to as the weighted Euclidean distance measure.

3. *Euclidean Distance:* The assumption here is exactly similar to the one above, except we further assume that all features have equal variance. Then $D_i(f)$ reduces to,

$$D_i(f) = (f-\bar{f}_i)\ (f-\bar{f}_i)^T \qquad (24)$$

which is referred to as the Euclidean distance measure.

4. *Correlation Distance:* The assumption here is exactly the same for the Euclidean distance measure. If we expand the expression, $D_i(f)$, for the Euclidean case we get

$$D_i(f) = f^2 - 2f\bar{f}_i + \bar{f}_i^2 \qquad (25)$$

Now if we normalize the middle term by dividing it with the individual variances of f and f_i, we obtain the correlation between the two vectors f and f_i. Since the first term of equation 25 is not a function of i and the last term is constant, $D_i(f)$ becomes:

$$D_i(f) = \frac{f\bar{f}_i}{(ff^T)^{1/2}\ (\bar{f}_i\bar{f}_i^T)^{1/2}} \qquad (26)$$

which is referred to as the correlation distance measure.

7.2.3 *Vowel Speech Data.*

The vowel speech data used to train the classifiers were obtained from five Arabic words representing the Arabic vowel system as listed in Table 10. Each Arabic utterance was repeated five times by three different speakers (2 male and one female). Recording was done in one session and was carried out in the ordinary office environment using the developed PC-based Arabic speech processing system described earlier.

7.2.4 *Feature Extraction.*

As stated earlier, the Arabic syllabic segmentation algorithm is used to locate syllable maxima which generally correspond to the steady state region of vowels. Once this is detected, three frames are used in the feature extraction stage, each of length 256 samples, weighted by a Hamming window.

Features extracted and used in the training and classification stages of the automatic Arabic vowel recognition system are listed in Section 7.2.1. The mathematical formulation behind these techniques can be found in reference [35].

/ ka ta ba /	كَتَبَ
/ ku ti ba /	كُتِبَ
/ ka: ta ba /	كاتَبَ
/ gi: la /	قِيلَ
/ gu:l /	قول

Table 10

Arabic Vowel Speech Data.

7.2.5 Software System Implementation.

Two software programs were written and compiled using TURBO PASCAL, one named "VO-rec1.pas" for automatic Arabic vowel recognition using test vowel data which is the same as training vowel data and the other named "VO-rec2.pas" for the recognition of test vowel data different than the training vowel data.

Program "VO-rec1.pas" proceeds by displaying a menu for selecting one parametric representation from the six mentioned in Section 7.2.1, then computes the recognition accuracy for all Arabic vowel classes using the Euclidean distance measure. The result is displayed in the form of a confusion matrix as shown in Table 11 for the three normalized formants as the set of parameters. The same is repeated for all distance measure techniques mentioned in Section 7.2.2.

Program "VO-rec2.pas" is similar to "VO-rec1.pas" except that it allows extraction of test vowel data from spoken Arabic utterances and then proceeds by giving the option of selecting one parametric representation. Finally, it displays the results in the form of a confusion matrix as shown in Table 12 for the Arabic vowel /a/ using the three normalized formants as features under four different distance measure techniques.

CONFUSION MATRIX

SPOKEN AS \ RECOGNIZED AS	a	a:	u	u:	i	i:	% Recognition
a	22	5	0	0	0	0	81.48
a:	0	27	0	0	0	0	100.00
u	0	0	27	0	0	0	100.00
u:	0	0	0	27	0	0	100.00
i	0	0	0	0	27	0	100.00
i:	0	0	0	0	5	22	81.48

Overall % Recognition = 93.83

Table 11

Confusion Matrix for the Arabic Vowel Classes.

CONFUSION MATRIX

RECOGNIZED AS

DIST_MEASURE	a	a:	u	u:	i	i:
EUCLIDEAN	3	0	0	0	0	0
CORRELATION	3	0	0	0	0	0
W_EUCLIDEAN	3	0	0	0	0	0
MAHALANOBIS	3	0	0	0	0	0

Table 12

Confusion Matrix for Arabic Vowel /a/

The main routines responsible for extracting the different parametric representations, and computing the results using the different distance measure techniques are given in Appendix E.

7.2.6 Results and Discussions.

In order to evaluate the performance of the Arabic vowel recognition system, two main investigations were carried out. The first investigation dealt with the suitability of the set of selected features relative to others mentioned before in terms of giving maximum separability among the different Arabic vowel classes. Scatter plots of the Arabic vowel system, using the different sets of features listed in Section 7.2.1, are shown in Figures 42 to 47. From these scatter plots it is noticed that the Arabic vowel classes (/a/, /a:/, /i/, /i:/, /u/ and /u:/) are quite distinct for most of the sets of features used. Hence, it is quite reasonable to expect the performance of the Arabic vowel recognition system to be high.

The second investigation is concerned with performing a comparative study of the Arabic vowel recognition process using different sets of features with different distance measure techniques. The performance of the recognition system is estimated under the following conditions:

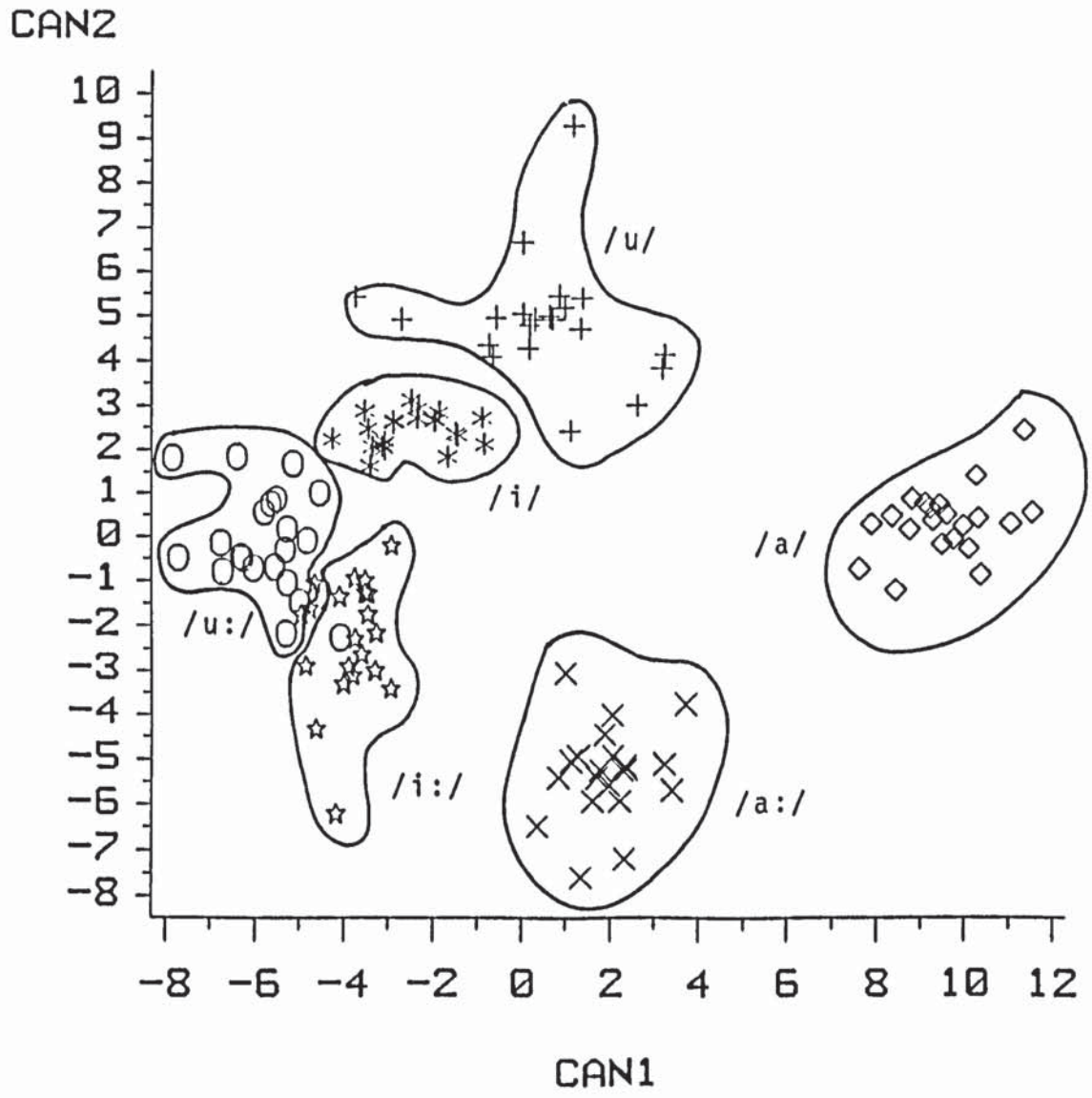


FIGURE 42

SCATTER PLOT OF ARABIC VOWEL USING FFT COEFFICIENTS AS FEATURES.

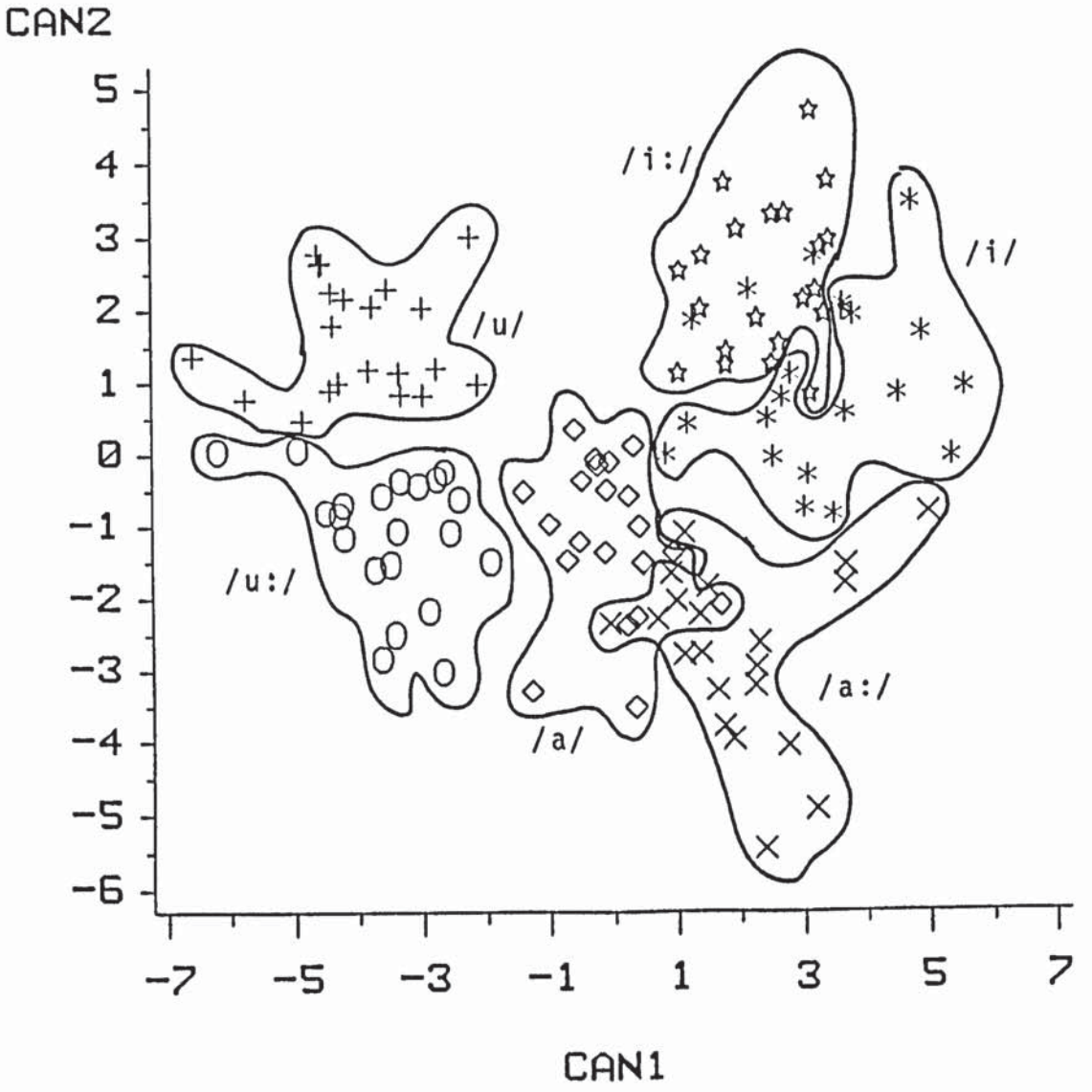


FIGURE 43

SCATTER PLOT OF ARABIC VOWEL USING CEPSTRUM COEFFICIENTS AS FEATURES.

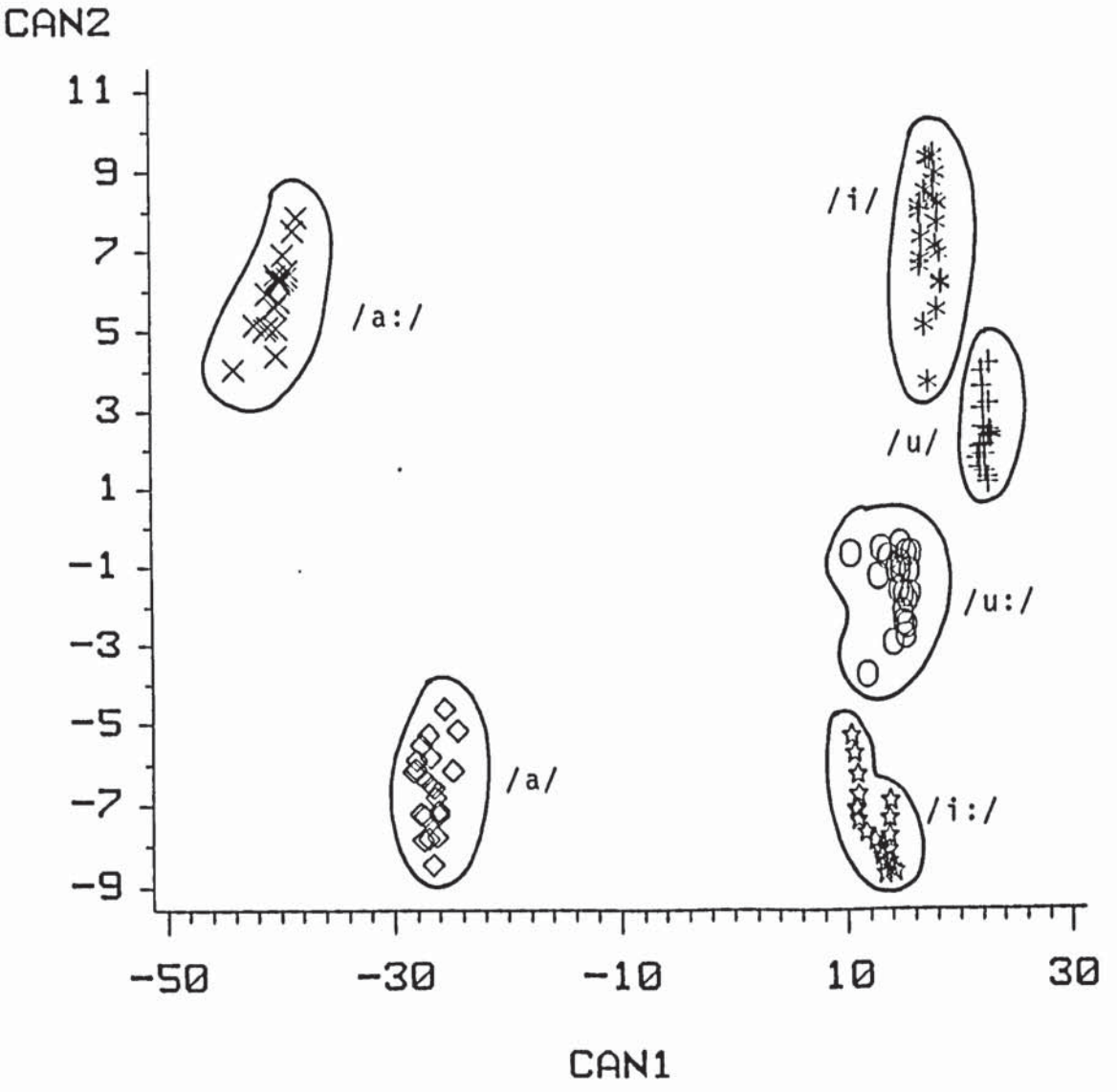


FIGURE 44

SCATTER PLOT OF ARABIC VOWEL USING AUTOCORRELATION COEFFICIENTS AS FEATURES.

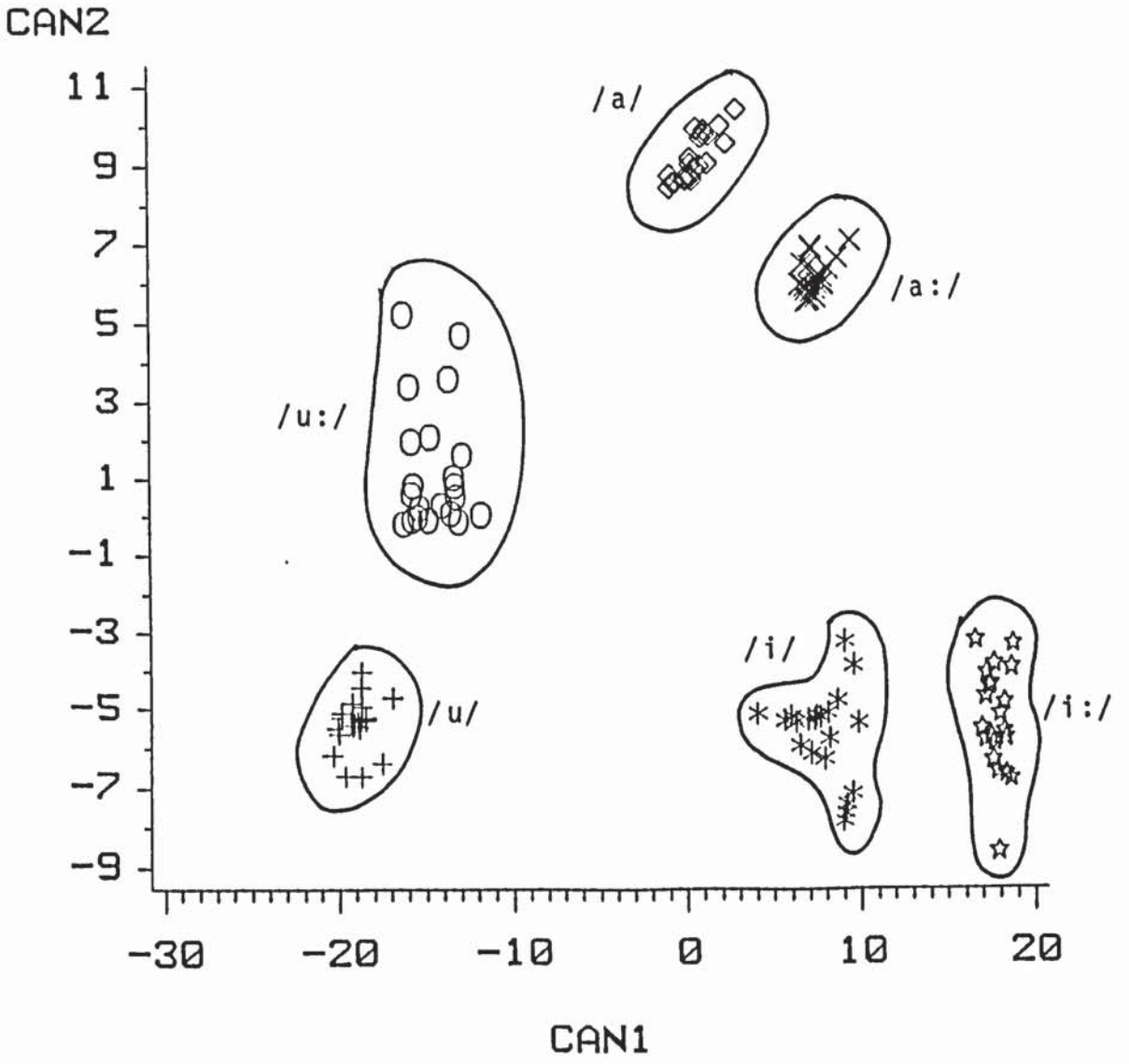


FIGURE 45

SCATTER PLOT OF ARABIC VOWEL USING LINEAR PREDICTION COEFFICIENTS AS FEATURES.

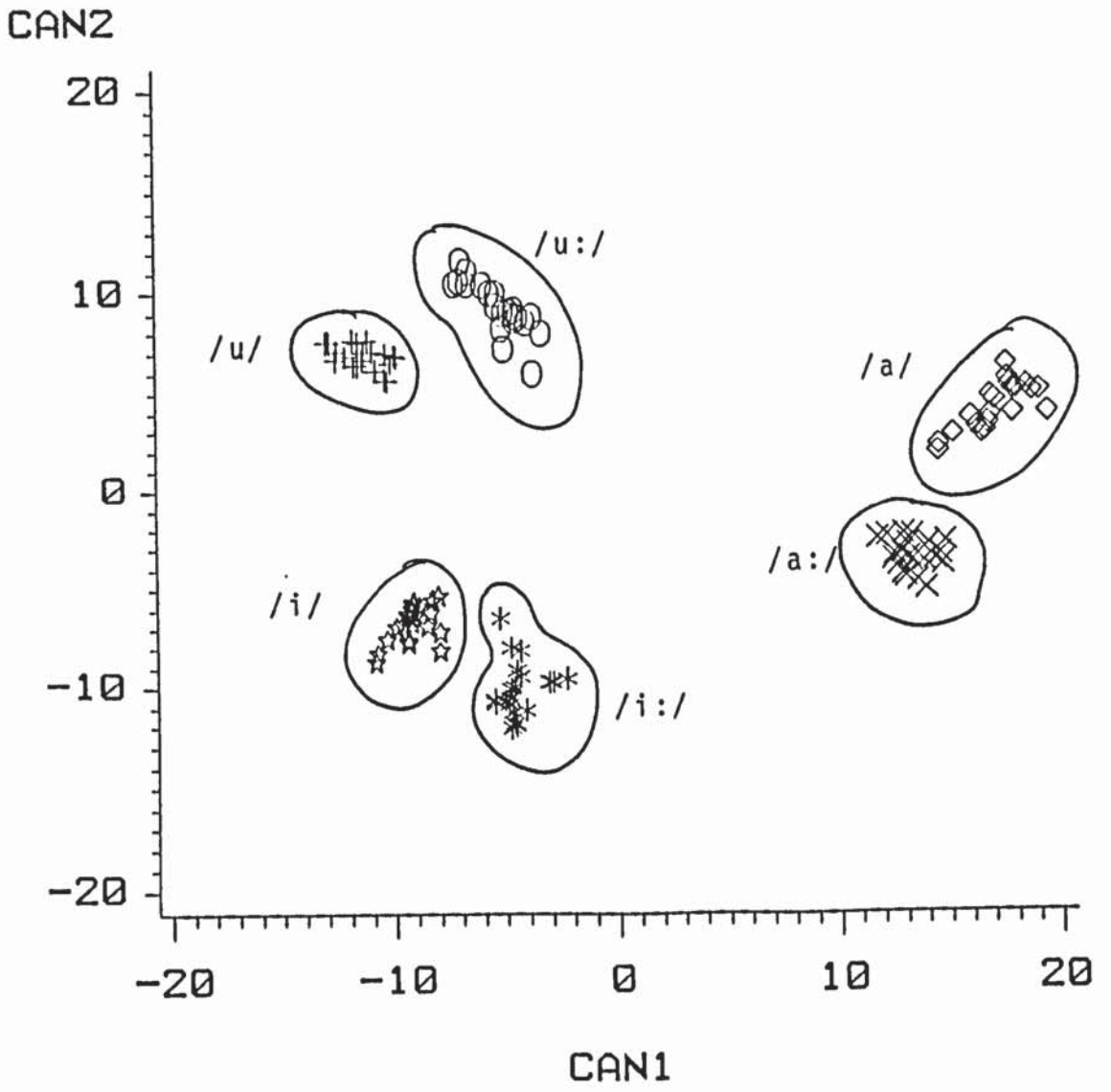


FIGURE 46

SCATTER PLOT OF ARABIC VOWEL USING REFLECTION COEFFICIENTS AS FEATURES.

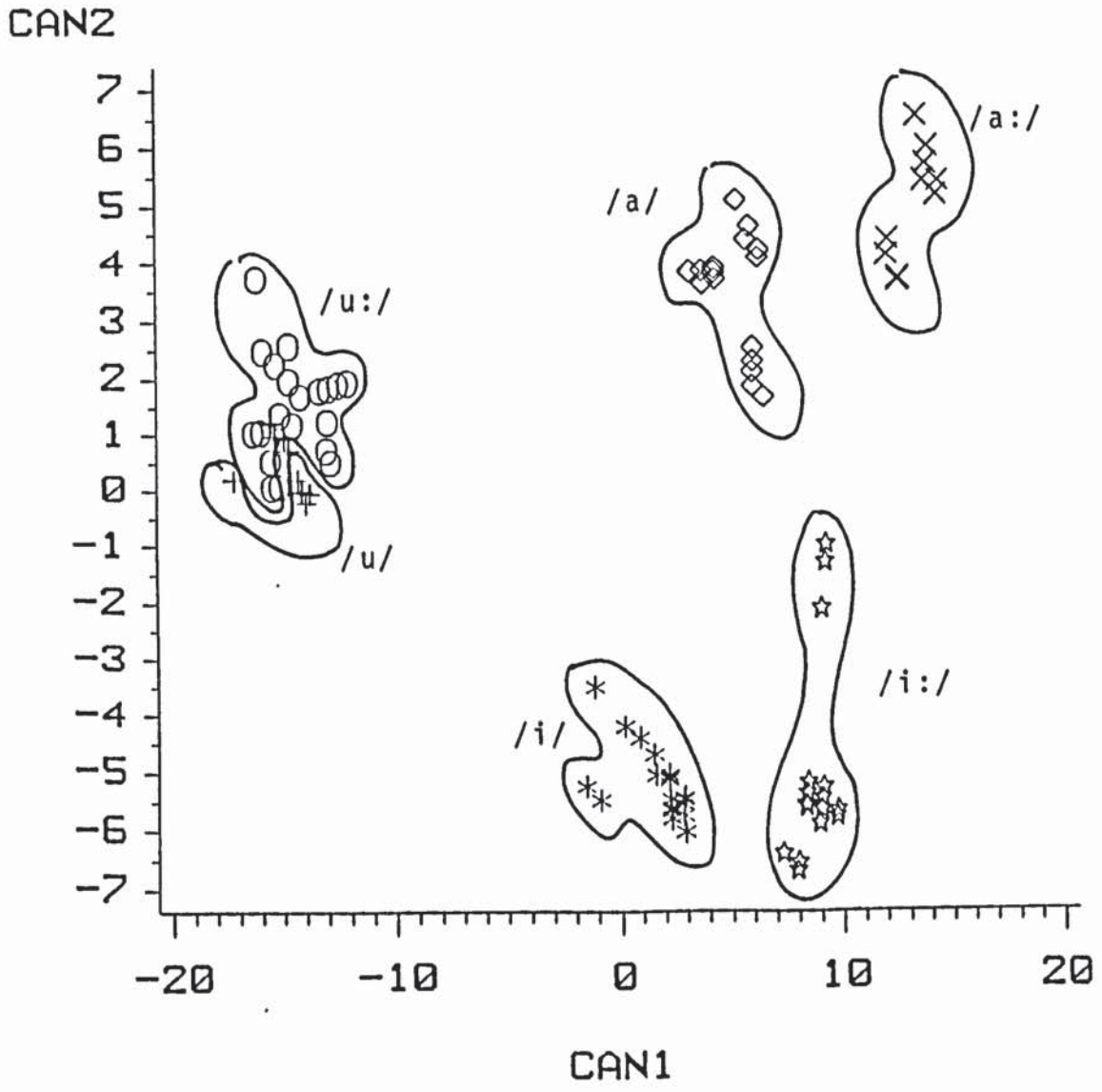


FIGURE 47

SCATTER PLOT OF ARABIC VOWEL USING FORMANTS COEFFICIENTS AS FEATURES.

1. The training and test data are the same. During the training phase all parameters required to design the classifier are computed, i.e., mean vectors and covariance matrices. This is done for all six Arabic vowel classes. The same training data are used as test data.

The recognition performance for each set of features used, and with different distance measures techniques is shown in Tables 13 to 18. The best performance was obtained when the reflection coefficients were used as a set of features (98.92%), followed by the normalized autocorrelation coefficients (97.34%), cepstrum coefficients, FFT coefficients, the first three normalized formants and the linear prediction coefficients. In all cases the best performance was achieved when the Mahalanobis distance measure technique is used. This is quite predictable since this distance measure requires both first and second order statistics. Euclidean and correlation distance measure techniques are simpler to implement and are faster to compute since they require only first order statistics. Recognition performance using these distance measures are almost equal. Arabic vowel recognition results, when using the reflection coefficients

Distance Measure	Recognition Accuracy of vowels, %						Total Recognition Accuracy (%)
	a	a:	u	u:	i	i:	
Euclidean	100	100	95	100	62.96	96.30	92.37
Correlation	100	100	95	100	62.96	100	92.99
Weighted Euclidean	100	100	98	100	98	74.07	95.01
Mahalanobis	100	100	74.07	100	100	100	95.68

Overall Accuracy = 94.01%

Table 13

Arabic Vowel Recognition Performance using 25 FFT Coefficients as the set of Features, and four Distance Measures Techniques.

Distance Measure	Recognition Accuracy of vowels, %						Total Recognition Accuracy (%)
	a	a:	u	u:	i	i:	
Euclidean	100	92.59	92.59	96.3	100	100	96.3
Correlation	100	92.59	92.59	96.3	100	100	96.3
Weighted Euclidean	100	92.59	92.59	96.3	100	100	96.3
Mahalanobis	100	100	100	96.3	100	100	98.77

Overall Accuracy = 96.91%

Table 14

Arabic Recognition Performance using 20 Cepstrum Coefficients as the set Features, and four Distance Measures Techniques.

Distance Measure	Recognition Accuracy of vowels, %						Total Recognition Accuracy (%)
	a	a:	u	u:	i	i:	
Euclidean	100	100	90	100	62.96	90	96.66
Correlation	100	100	90	100	62.96	90	96.66
Weighted Euclidean	100	100	90	100	98	90	96.66
Mahalanobis	100	100	96.3	100	100	100	99.38

Overall Accuracy = 97.34%

Table 15

Arabic Vowel Recognition Performance using Auto-Correlation Coefficients as the set of Featrures, and four Distance Measures Techniques.

Distance Measure	Recognition Accuracy of vowels, %						Total Recognition Accuracy (%)
	a	a:	u	u:	i	i:	
Euclidean	100	92.59	96.3	100	100	100	98.15
Correlation	100	92.59	96.3	100	100	100	98.15
Weighted Euclidean	100	100	100	100	96.3	100	99.38
Mahalanobis	100	100	100	100	100	100	100

Overall Accuracy = 98.92%

Table 16

Arabic Vowel Recognition Performance using 14 Reflection Coefficients as the set of Features, and four Distance Measures Techniques.

Distance Measure	Recognition Accuracy of vowels, %						Total Recognition Accuracy (%)
	a	a:	u	u:	i	i:	
Euclidean	100	100	96.3	85.19	51.85	62.96	82.72
Correlation	100	100	100	96.3	55.56	62.96	85.80
Weighted Euclidean	100	100	96.3	96.3	62.96	62.96	86.42
Mahalanobis	100	100	100	100	59.26	62.96	87.04

Overall Accuracy = 85.50%

Table 17

Arabic Vowel Recognition Performance using 14 LP Coefficiets as the set of Features, and four Distance Measures Techniques.

Distance Measure	Recognition Accuracy of vowels, %						Total Recognition Accuracy (%)
	a	a:	u	u:	i	i:	
Euclidean	81.48	100	100	100	100	81.48	93.83
Correlation	48.15	100	100	88.89	100	81.48	86.42
Weighted Euclidean	100	100	100	100	88.89	74.07	93.83
Mahalanobis	100	100	100	100	62.96	100	93.83

Overall Accuracy = 94.01%

Table 18

Arabic Vowel Recognition Performance using 3 Normalized Formants Coefficients as the set of Features, and four Distance Measures Techniques.

as a set of features with Mahalanobis distance measures, are shown in Table 19 in the form of a confusion matrix. It is seen that recognition errors are due to confusion with neighbouring vowels.

2. The training Arabic vowel data and test data are different. First the recognition system is trained with 27 samples per Arabic vowel class. A total of 27 x 6 samples are used in the training process. Then 27 x 6 different test data are applied. Table 20 shows the performance of the Arabic vowel recognition system using six different parametric representations and four distance measure techniques. Here we notice that the best performance is obtained when using cepstrum coefficients and Mahalanobis distance measures.

Overall the most appropriate parametric representation for the Arabic vowel classes would be the reflection coefficients because they are relatively easy to compute and normalization is inherent in their computation. They also represent a meaningful model of speech production.

CONFUSION MATRIX Recognized as							% Recognition
	a	a:	u	u:	i	i:	
a	27	0	0	0	0	0	100%
a:	0	27	0	0	0	0	100%
u	0	0	27	0	0	0	100%
u:	0	0	0	27	0	0	100%
i	0	0	0	0	16	11	59.26%
i:	0	0	0	0	10	17	62.96%
						Overall	87.04%

Table 19

Confusion Matrix for the Recognition of 27 Samples of Arabic Vowels using the Reflection Coefficients as a set of Features and with Mahanabolis Distance Measure Technique.

Parametric Representation	Euclidean Distance Measure	Correlation Distance Measure	Weighted Euclidean Distance Measure	Mahalanobis Distance Measure
1. FFT Coefficients	72.1	72.0	72.5	75.0
2. Cepstrum Coefficients	80.8	80.8	81.4	86.0
3. Normalized Auto-Correlation Coefficients	75.1	75.2	77.4	82.0
4. Reflection Coefficients	71.0	71.0	72.6	79.2
5. Linear Prediction Coefficients	61.1	61.0	61.0	70.0
6. Normalized Formants	55.8	55.8	55.6	73.0

Table 20

Arabic vowel Recognition Performance using Different Parametric Representations and four Different Distance Measures Techniques.

7.2.7 *Conclusion.*

Six different parametric representations of Arabic vowel steady state regions were investigated with the objective of performing a comparative study of the performance of an Arabic vowel recognition system, under different distance measures techniques. The best performance of the Arabic vowel recognition system was achieved when using the cepstrum coefficients for parametric representation with Mahalanobis distance measure technique.

7.3 *Isolated Arabic Word Recognition System*

7.3.1 *Introduction.*

Many isolated commercial and experimental word recognition systems exist today [57]. This is due to the simple fact that it is much easier to tackle the speech recognition problem at the word level rather than the sub-word level, which is much more susceptible to context variations. The main disadvantages are that, when dealing with a very large size recognition vocabulary, the scanning of reference templates to find the best match can be very time-consuming. A study by Shipman and Zue [58] suggests that, using partial phonetic description of sounds, one is able to eliminate all but a handful of word candidates from a large size recognition vocabulary.

The second application of the Arabic syllabic segmentation algorithm is in the implementation of an isolated Arabic word recognition system. Syllabic units obtained by applying the syllabic segmentation algorithm are advantageous to have for the following reasons. First, we can have a reduced set of templates depending on the vocabulary. Second, for limited size recognition vocabulary, one can use stressed syllables as the primary units for speech recognition thus shorting the time for recognition as only one stressed syllable per word needs to be matched.

7.3.2 Speech Recognition Problem and the Proposed Recognition Strategy for Arabic.

Every speech recognition system available today whether commercial or experimental, perform several tasks which are sensitive to a certain limit due to the high degree of spectral variability and noise problems. Therefore, more robust recognition algorithms are needed to handle these problems. Of the two speech recognition strategies mentioned earlier (Section 3.2) the approach suggested here for an Arabic speech recognition system is to use a combined strategy of an acoustic-phonetic decoding into syllabic units and statistical pattern recognition as was shown in Figure 1. The important step here is the utilization of the developed Arabic

syllabic segmentation algorithm, which is also speaker independent. Once these syllabic units are obtained, appropriate feature extraction techniques are applied to generate a library of reference templates corresponding to the recognition vocabulary. Speech templates are normally generated during the training phase of the recognition process.

7.3.3 *Stressed Syllables as Primary Units for Speech Recognition.*

It is a well known fact, that not all information is of equal importance in speech recognition. It is therefore necessary to identify the important information and to develop a recognition algorithm which functions by maximizing this information. The sounds of unstressed Arabic syllables are much more variable than those in stressed syllables. For instance, the variations in the pronunciation of the Arabic word /jatma.a/ shown in Figures 33 and 34 all occurred in the unstressed syllables. Results have also shown that more reliable acoustic cues for the perception of phonemes exist in stressed syllables [59]. Since the information in stressed syllables seems to be more salient, and because of the characteristics of the Arabic language in terms of its syllabic structure and lexical stress rules the stressed Arabic syllables are used as the primary speech recognition

units for Arabic digits. The isolated Arabic digit speech recognition system is shown in Figure 48.

7.3.4 *Speech Input and End Point Detection.*

The PC-based speech processing system (Chapter 4) is used to input and digitize utterances. Digital speech signals are stored in a temporary buffer pending further processing. End point detection is performed manually, by placing the cursor markers for the start and the end of the displayed speech pressure waveforms, on the color screen monitor, as shown in Figure 49.

7.3.5 *Syllabic Segmentation and Feature Extraction.*

Once the acoustical speech waveform is entered, digitized and the end-point detection performed, the syllabic segmentation algorithm explained in Chapter 6 is invoked. The result from the application of this algorithm is a set of segments corresponding to the phonetic syllabic units present in the utterance. One of these segments is identified to be the stressed syllable by comparing it to the rest of the other segments in terms of its intensity and duration values. The other segments represent the unstressed syllables. The next step is to extract a set of numerical features that adequately represent the speech signals. Various numerical features have been reported [35]. The numerical feature extraction

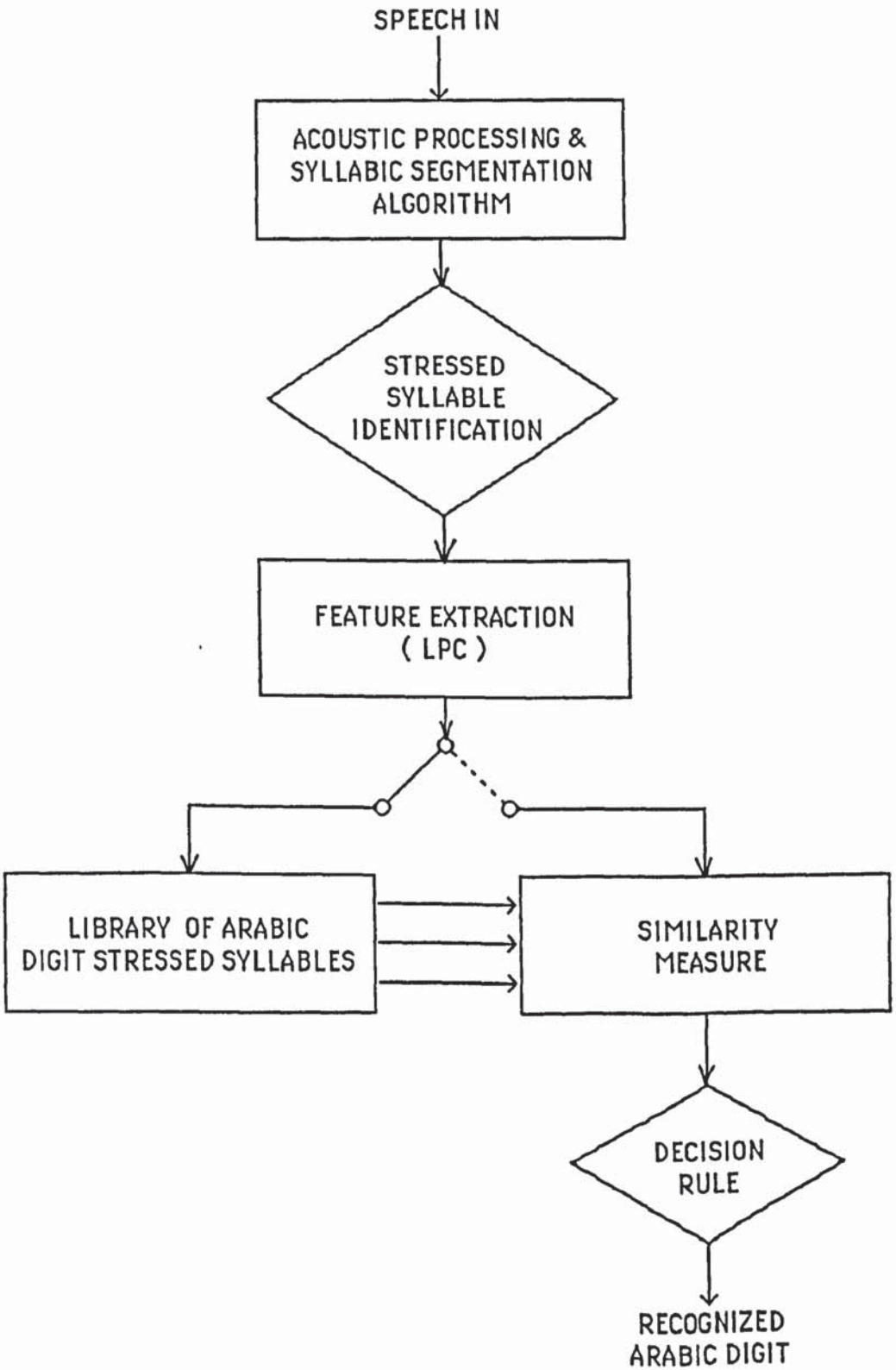


FIGURE 48

ARABIC DIGIT RECOGNITION SYSTEM.

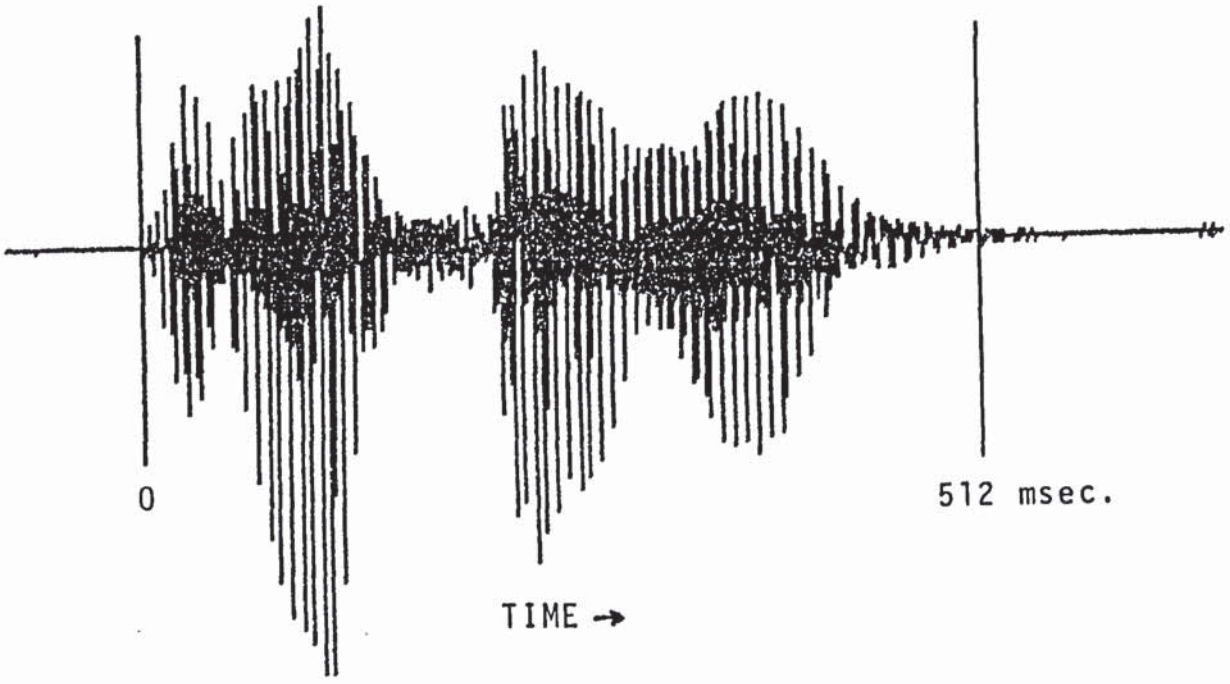


FIGURE 49

CURSOR MARKERS FOR START AND END POINTS OF THE SPEECH PRESSURE WAVEFORM.

stage can be done either in the time domain, or in the frequency domain. Examples of types of numerical features used in speech recognition were mentioned in Chapter 3. The choice of one feature type over the other depends on many factors such as ease of computation and how well it represents the speech signals temporal behaviour. LPC coefficients were chosen as the set of the numerical features representing the syllabic segments for the following reasons:

1. They can be obtained fairly easily through computationally efficient algorithms [60].
2. They can be used to obtain spectral information such as formants and spectral energy [12].
3. The most important fact is that they are meaningful to the model of the speech production process.

7.3.6 Linear Prediction Analysis of the Arabic Syllabic Segments.

Linear prediction (LP) analysis is carried out on each Arabic stressed syllabic segment. The objective is to determine the set of values of the LP coefficients $a_1 \ldots a_p$ in each time frame. The LP coefficients represent the filter coefficients in an all pole model for speech production. The transfer function H(z) of this model is given in the equation

below:

$$H(z) = \frac{G}{1 + \sum_{i=1}^{P} a_i z^{-i}} \qquad (27)$$

where,

G equals the gain (magnitude of periodic source),
P is the number of poles in the digital filter, and
a_i are the filter coefficients (LP coefficients).

For accurate modelling of the speech spectrum, a high order of LPC (Linear Prediction Coding) is required. Different orders of LPC were used in speech recognition. For example, Rabiner [61] used an 8 pole model, while Davis [62] used a 10 pole model. As a rule of thumb, a good approximation is to use one pole pair for every 1 kHz, that is, for speech bandwidth of 5000 Hz, 10 poles is a good representation. 14 poles were used for the speech signals which are sampled at 10 kHz rate.

The way to determine the LPC coefficients $a_1 \ldots a_p$, is by minimizing the error $e(n)$ between the actual and predicted speech samples in an optimal sense. An optimal way is to minimize the total square error $\{e(n)\}^2$ over the frame size as given in equation 28 below:

$$\sum_{n=0}^{N-1} \{e(n)\}^2 = \sum_{n=0}^{N-1} \{x(n) - \sum_{i=1}^{P} a_i \, x\,(n-i)\}^2 \qquad (28)$$

The minimization procedure is fully described by Witten [63], and is commenced by differentiating equation 28 with respect to each LP coefficient in order to arrive at a set of p equation with p unknown LP coefficients. The equations put in a matrix form are as follows:

$$\begin{vmatrix} R(0) & R(1) & \cdots & R(p-1) \\ R(1) & R(0) & \cdots & R(p-2) \\ \vdots & \vdots & \cdots & \vdots \\ R(p-1) & R(p-2) & \cdots & R(0) \end{vmatrix} \begin{vmatrix} a(1) \\ a(2) \\ \vdots \\ a(p) \end{vmatrix} = - \begin{vmatrix} R(1) \\ R(2) \\ \vdots \\ R(p) \end{vmatrix}$$

where,

R(1) represents the first autocorrelation.

The set of p equations are solved simultaneously in order to arrive at the estimates $a_1 \ldots a_p$. Two methods are used for the solution of these equations, namely, the autocovariance method, and the autocorrelation method which are described by Witten [63]. The autocorrelation method was used to obtain the LP coefficients through the implementation of the Levinson and Durbin efficient recursive algorithm [60]. The algorithm proceeds as follows:

$$E_0 = R(0) \tag{29}$$

For $j = 1, 2, \ldots p$

$$K_j = -\{R(j) + \sum_{k=1}^{j-1} a_{j-1}(k)\, R(j-k)\} \,/\, E_{j-1}$$

$$a_j(j) = K_j \tag{30}$$

$$a_j(k) = a_{j-1}(k) + K_j a_{j-1}(j-k), \quad 1 \leq k \leq j-1$$

$$E_j = (1-K_j^2)\, E_{j-1}$$

The final solution of this algorithm is given by the coefficients $a_p(k)$, $1 \leq k \leq p$. The intermediate quantities K_j are known as the reflection coefficients. The complete PASCAL algorithm used for the computation of the LP coefficients is shown in Appendix C.

7.3.7 *Time Normalization.*

To compare two acoustic patterns, some form of time normalization of the time-varying features within each acoustic pattern has to be brought about. Comparisons using linear time alignment of the two acoustic patterns have been known to give inaccurate results because linear mapping implies that certain temporal acoustical events have been either compressed or expanded in a way that does not hold any relation to the matched acoustical event.

Time normalization is now achieved by a process known as "Dynamic Time Warping" (DTW), developed by

Itakura [64]. The application of this technique to isolated and connected word recognition systems has improved the recognition accuracy of these systems. The DTW technique is described below.

7.3.8 *Dynamic Time Warping (DTW).*

DTW is a method of non-linear time alignment between two acoustical events, i.e., the test and the reference patterns. The test pattern is compared to each reference pattern and the score of a cost function is computed, based on finding an optimal path between the test and the reference patterns. Optimality of the path is based on finding the least accumulated distance measure from each matching point on the grid that contains the test and reference pattern points. Templates or reference patterns are normally generated during the training phase of the recognition process. Generally they represent a set of vectors, each vector represents certain acoustical events in a time frame of 10-30 msec. in duration. Each utterance is represented by a sequence of these vectors. In our case, the acoustical patterns represent the Arabic stressed syllabic units, the frame size is 256 speech samples (25.6 msec.), and each vector is made of 14 LP coefficients. The following mathematical treatment is based essentially on the paper of Sakoe and Chiba [65].

Let,

R_i - R(1), ..., R(N) be a reference word i,

T - T(1), ..., T(M) be a test utterance,

d(i, j) - ||R(i) - T(j)|| denotes a distance between frame i of R, and frame j of T, and let C be a parametric curve of the plane defined by C(k) - [i(k), j(k)], k-1, ... k, where C(1) - point (1, 1), and C(k) - point (N, M) as shown in Figure 50. The curve C is called the "Warping Function". The time normalized distance between speech patterns R and T is defined by:

$$D(R, T) = Min_C \frac{\sum_{k=1}^{K} d\{c(k)\}\, w(k)}{\sum_{k=1}^{K} w(k)} \qquad (31)$$

where,

w(k) are weighting coefficients, and

K - number of points on the warping function c.

Searching for the optimal warping function can be time consuming, therefore some restrictions are applied aimed at restricting the domain of the mapping space. The most common constrains applied, to the mapping function are:

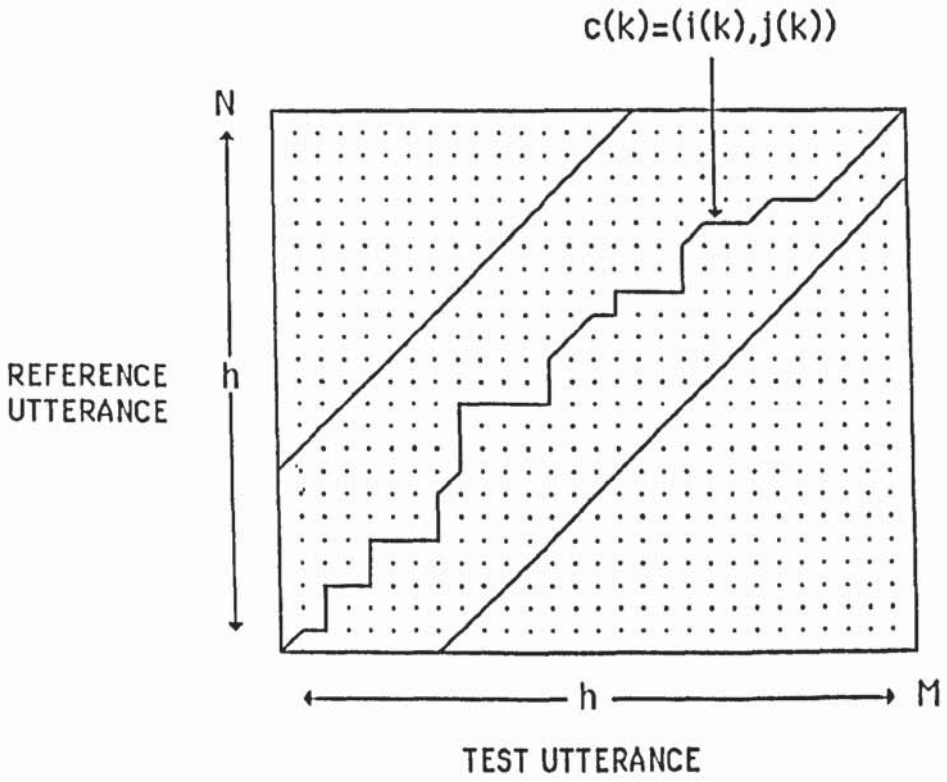


FIGURE 50

WARPING FUNCTION.

1. The function must match the end points, i.e., C(1) = point (1, 1) and C(k) = point (N, M).
2. The function must be monotonically increasing thus achieving a time ordering preservation, i.e., $C(j-1) \leqslant C(j)$.
3. The function must be continuous i.e., $C(n) - C(n-1) \leqslant 1$.
4. The function must not allow for too excessive expansion or compression in the time scale. Therefore the path of the mapping space is restricted to exclude all points that may provide excessive time difference, i.e., $|i(k)-j(k)| \leqslant h$, where h is a positive integer called "Window Length".
5. The gradient of the warping function must not be allowed to be too steep or too gentle, so as to avoid undesirable time-axis warping. Various schemes both symmetric and asymmetric are suggested by Sakoe and Chiba [65] for the warping function. Examples of these schemes are shown in Figure 51. If the warping function C(k) is allowed to step forward m-times in the horizontal or vertical position, then it is not allowed to step any further in the same direction before stepping at least n-times in the diagonal direction. The slope P is defined as n/m. According to Sakoe and Chiba [65], the best performance is achieved

when the slope P is equal to one. The scheme shown in Figure 52 was used. It has a slope P equal to 1.

The chosen warping function scheme shown in Figure 52 specifies three different ways to reach the point (n, m) as indicated by the arrows. If the best accumulated distances at points (n-2, m-1), (n-1, m-1), and (n-1, m-2) are D(n-2, m-1), D(n-1, m-1), and D(n-1, m-2), respectively, then the best accumulated distance to reach the point (n, m) is equal to D(n, m) as given below.

$$D(n, m) = Min_{c}\begin{cases} D(n-2, m-1) + rd(n-1, m) + d(n, m) \\ D(n-1, m-1) + rd(n, m) \\ D(n-1, m-2) + rd(n, m-1) + d(n, m) \end{cases} \quad (32)$$

r is a weighting coefficient and is normally given the value two. Therefore, the distance at each allowed point in the n x m grid of points is evaluated recurrently in an ascending order with respect to the reference and test utterance coordinates, until the point (n, m) is reached. The final time-normalized distance between the reference and test utterance is given by the equation below.

$$D(R, T) = \frac{D(N, M)}{M} \quad (33)$$

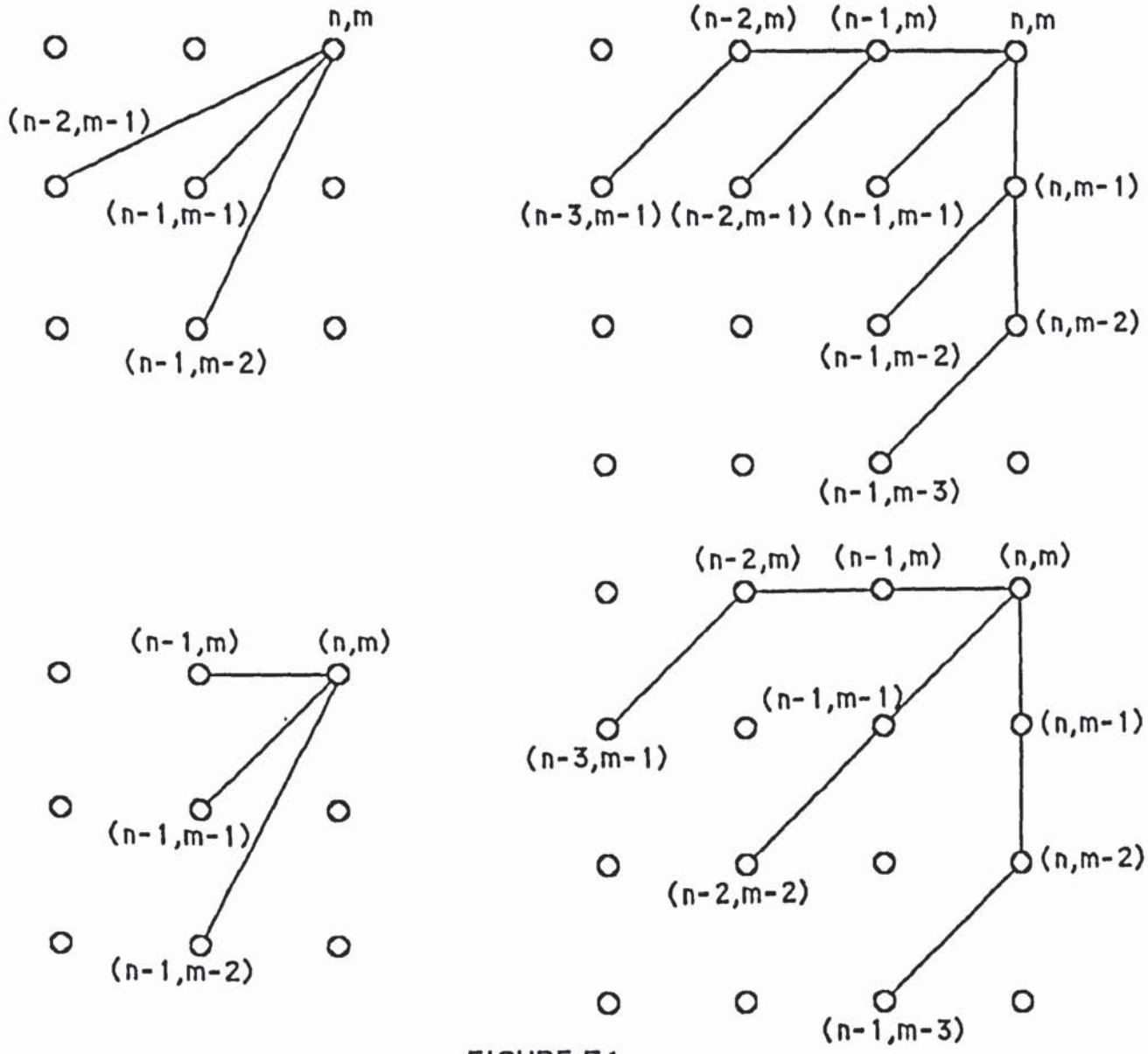


FIGURE 51

VARIOUS WARPING SCHEMES

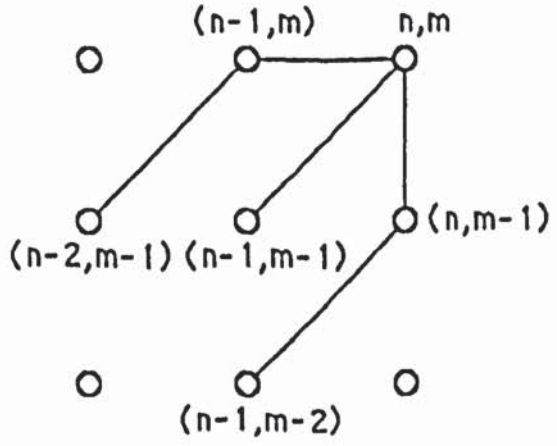


FIGURE 52

SYMMETIC WARPING SCHEME

7.3.9 Distance Measure.

The most important step in the dynamic-programming, time-warp algorithm is the distance or similarity measure between the test and reference frames. A good distance measure scheme should have the following features:

1. Symmetric, i.e., $d(x, y) = d(y, x)$.
2. Positive definite, i.e., $d(x, y) \neq 0$ if $x \neq y$ and $d(x, x) = 0$.
3. Satisfy the triangle rule, i.e., $d(x, y) \leq d(x, z) + d(y, z)$.
4. Should have a physical meaningful interpretation.
5. Can be efficiently computed.

Most of the distance measures proposed in the literature [66] have led to high recognition rates. However, they do not satisfy all the previous conditions. The choice of one distance (similarity) measure against another is influenced by the feature set used. For example, using the cepstral coefficients as features, then the most appropriate distance measure is the Euclidean distance.

In our case, since LP coefficients were used as the set of features representing each frame of Arabic stressed syllabic segment, the most appropriate measure of similarity between the test and template frames is the log likelihood ratio as proposed by

Itakura [64]. However in this implementation, the log was not taken. This distance measure is given in the equation below:

$$d = \frac{a_p^T \, R_p \, a_p}{a_s^T \, R_p \, a_s} \qquad (34)$$

where,

a - vector of LP coefficients of given frame,

R - matrix of autocorrelation coefficients,

T denotes transpose,

p denotes template frames, and

s denotes test frames.

7.3.10 Software System Implementation.

A software program named "IAWRS.PAS" was written and compiled using TURBO PASCAL. The program proceeds by displaying a menu for selecting either of two modes, i.e., training mode or recognition mode.

During the training mode, stressed syllabic units are extracted and their LPC representations are stored in appropriate files as speech templates. Extraction of syllabic units are facilitated through the application of the Arabic syllabic segmentation algorithm.

During the recognition mode, an unknown test utterance, represented by its stressed syllabic unit, is compared against all templates using the "DTW"

technique. Comparison results are presented in terms of scores. An option is also given to display the warping path as shown in Figure 53 for the test Arabic digit 6 against template digit 9.

The main routines such as "DTW" are listed in Appendix F.

7.3.11 *Experimental Conditions.*

The PC-based speech processing system was used to enter and store spoken utterances. The Arabic digits vocabulary was used to test the performance of the isolated Arabic word recognition system. Each Arabic digit was uttered five times by two male speakers referred to as speaker A and B, and one female speaker referred to as speaker C. All speech recordings were done in the normal office environment.

7.3.12 *Arabic Digits Recognition Results.*

The syllabic structures of the Arabic digits are shown in Table 21. Parameters used to represent the stressed Arabic syllabic units were the linear prediction (LP) coefficients. Stressed syllabic units are identified by their energy and duration values, i.e., stressed syllabic units have more energy and longer duration than non-stressed syllabic units. For each speech frame (256 samples) 14 LP and autocorrelation coefficients are extracted and stored. During the recognition phase, frames

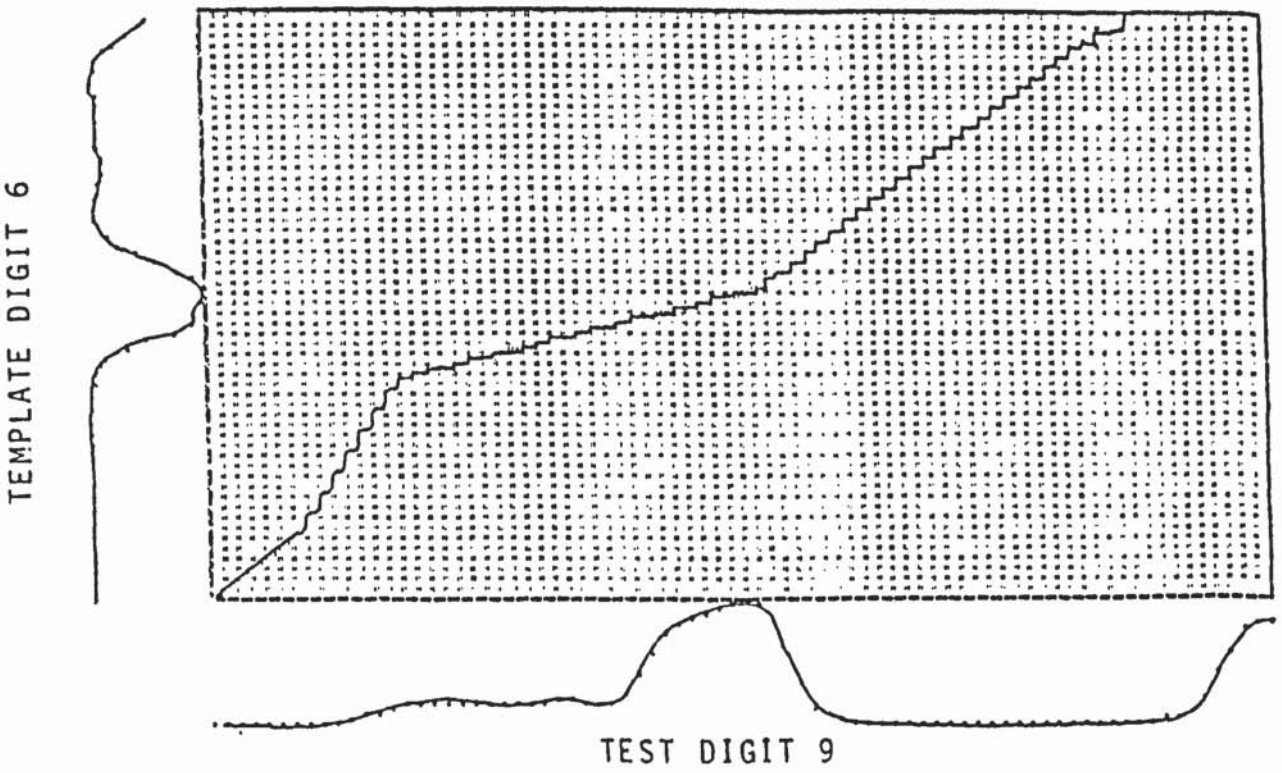


FIGURE 53

ENERGY PROFILE AND WARPING PATH FOR THE ARABIC TEST DIGIT 9 AGAINST TEMPLATE DIGIT 6.

Arabic Digit		Syllabic Structure
صفر	0	/sə/ /fir/
واحد	1	/wə:/ /ħid/
اثنين	2	/ʔəθ/ /nəjn/
ثلاثه	3	/θə/ /lə:/ /θəh/
اربعه	4	/ʔər/ /bə/ /ɣəh/
خمسه	5	/xəm/ /səh/
سته	6	/sit/ /təh/
سبعه	7	/səb/ /ɣəh/
ثمانيه	8	/θə/ /mə:/ /ni/ /jəh/
تسعه	9	/tis/ /ɣəh/
عشره	10	/ɣə/ /ʃə/ /rəh/

Table 21

Syllabic Structure of Arabic Digits.

corresponding to the unknown syllabic segment are compared to each frame template using a similarity measure as given in equation 34.

This frame similarity measure is used in conjunction with the dynamic time-warping technique described earlier to calculate the minimum distance between the test and template stressed syllabic units. The following DTW constraint conditions were used:

1. Symmetric form for the warping function as shown in Figure 47.
2. The least accumulated distance, equation 29, was computed using a weight coefficient value of 1.5.
3. Variable size "Window Length" was used, which limits the path of the mapping space. An upper limit of size 30 was used, which corresponds to a 19.2 msec. time difference.

The performance of the isolated Arabic digit recognition system was tested in the following manner:

Test 1:

- Unit of Recognition: Stressed Arabic Syllabic Units.
- Recognition Mode: Speaker-dependent

Three speakers were used to supply the test utterances and templates for the spoken Arabic digit words. Five word utterances were recorded for each speaker for each of the digits 0 - 10, giving a total of 5 x 11 = 55 utterances per speaker. Recognition tests were carried out on each speaker in turn. In each case the first utterance for each digit word were used as templates and the other four utterances for each digit were tested against the templates for recognition. The tests were then repeated using the second, third, forth and five utterances in turn as templates, the other utterances being tested against the template. Thus 5 x 4 = 20 tests were made for each digit for each speaker in turn. The performances of the isolated Arabic digit recognition system for each speaker are shown in Tables 22, 23 and 24 for the respective speakers A, B and C. It is seen that the overall recognition accuracy for speakers A, B and C were 95%, 96% and 68%, respectively.

The performance of the isolated Arabic digit recognition system was also tested using LPC representation of whole Arabic digit durations instead of its LPC representations of stressed syllabic units. The performance of the Arabic digit recognizer increased by 2% at a cost of an increased processing time in the case of speaker B. This was

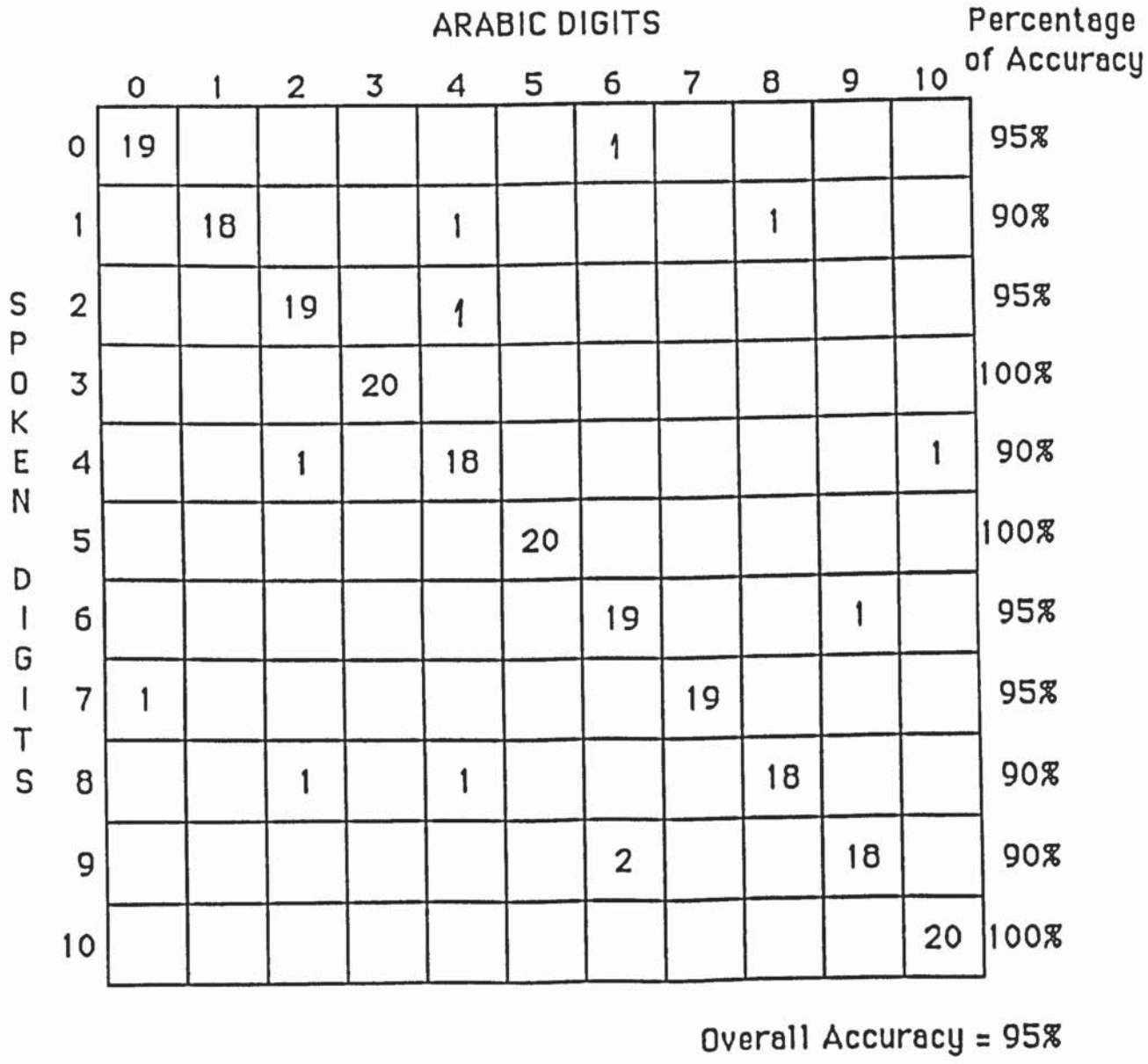


Table 22

Confusion Matrix for Recognition of
Isolated Arabic Digits for speaker A.

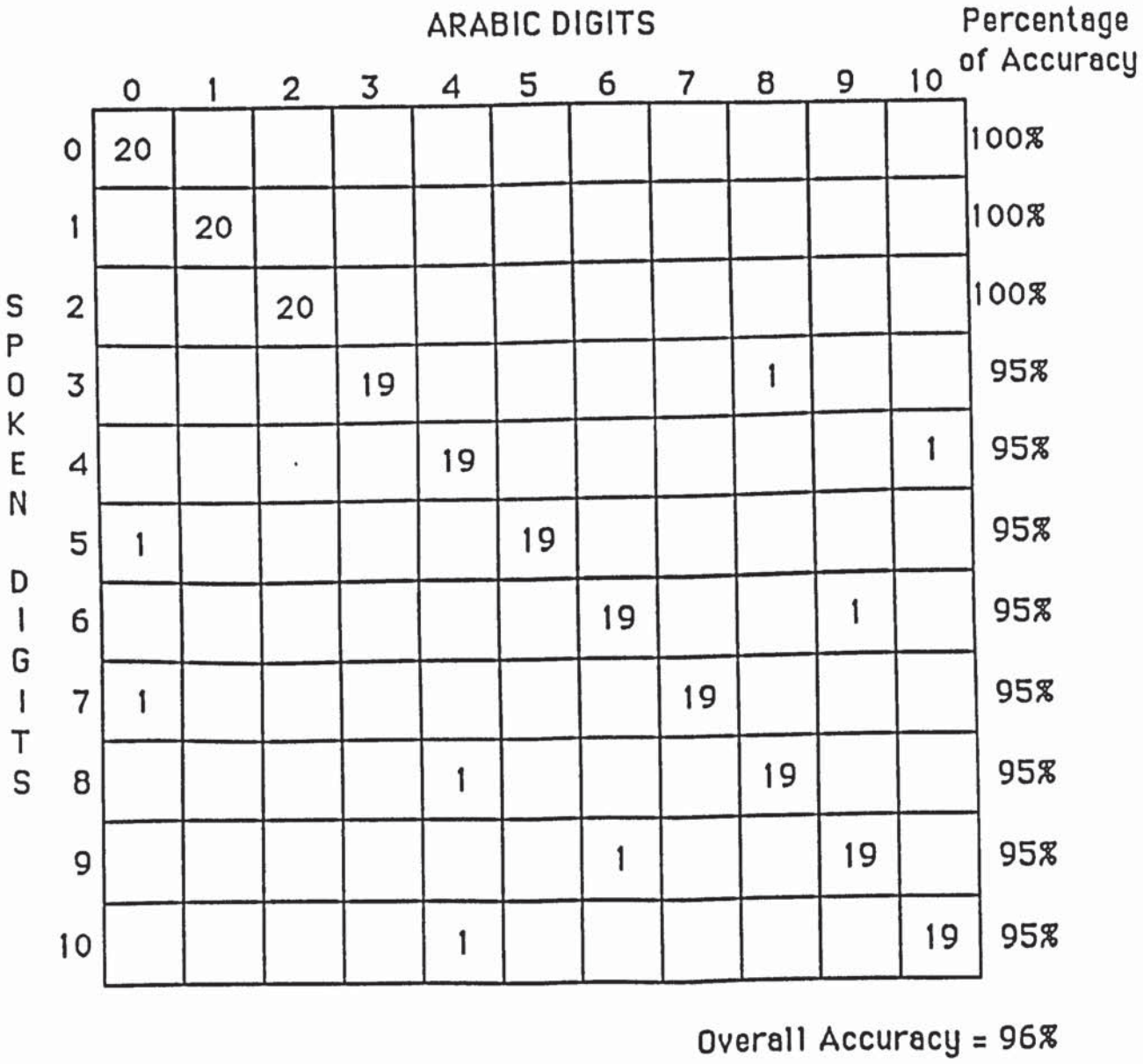


Table 23

Confusion Matrix for Recognition of
Isolated Arabic Digits for speaker B.

SPOKEN DIGITS \ ARABIC DIGITS	0	1	2	3	4	5	6	7	8	9	10	Percentage of Accuracy
0	16	1		1		1			1			80%
1		14		1	1				2	1	1	70%
2			17	2					1			85%
3		1		15			1	1	1	1		75%
4		1	2		12		1		3		1	60%
5		1		1		15	2			1		75%
6				1		3	12	2		2		60%
7	1			1		1	1	13		3		65%
8	1			1	1	2	1		11	1	2	55%
9						1	2	3		14		70%
10	2				1		1	1	3	1	11	55%

Overall Accuracy = 68%

Table 24

Confusion Matrix for Recognition of
Isolated Arabic Digits for speaker C.

achieved by modifying the main software routines responsible for generating Arabic speech digit templates.

Test 2:

- Unit of Speech Recognition: Stressed Arabic Syllabic Units.
- Recognition Mode: Speaker-independent

Two different sets of test conditions were used in this mode. In the first set of conditions the five Arabic digit utterances per Arabic digit of speaker A were used as templates while the 5 word Arabic digit utterances per Arabic digit of speakers B and C were used to supply the unknown test digits. The process was then repeated, but this time the Arabic digit utterances of speaker B were used as templates and those of speaker A and C as test digits. Thus, the total number of templates and test digits used in each case were 5 x 11 = 55 (one speaker) and 2 x 5 x 11 =110 (two speakers), respectively. The performance of the isolated Arabic digit system under these test conditions was very poor, being litter better than random

In the second set of test conditions, three samples from each of the five Arabic word utterances per Arabic digit per speaker were used as reference

templates, i.e., a total of 3 (digit utterances) x 3 (speakers) = 9 templates per digit were used. The remaining two digit utterances per speaker were used to supply the unknown test digits. Six different combinations of the three samples out of five Arabic word utterances per digit per speaker were used, giving us a total of 6 (combination) x 2 (remaining test sample occurrences) x 3 (number of speakers) = 36 Arabic text digits. The performance of the isolated Arabic digit recognition system was again very poor, again being little better than random.

The poor performance of the isolated Arabic digit recognition system using templates from a different speaker to the one being tested can be attributed to the different durations and the spectral variations among the different speakers when articulating the Arabic digits.

It is evident from Test 1 and 2 that the isolated Arabic digit recognition system is speaker-dependent.

7.3.13 *Conclusion.*

Successful implementation of an isolated Arabic word recognition system has been demonstrated for speaker-dependent modes. Preliminary evaluations of stressed syllabic segments as the primary units for speech recognition holds considerable promise for limited vocabulary speech recognition systems.

However, additional work needs to be done before a practical system can be implemented.

The results which are presented here should be taken as indicative of the power of the approach, but not as a final definitive performance figure.

7.4 A Module for Acoustic-Phonetic (Syllabic) Transcription of Arabic Speech

7.4.1 Introduction.

It has long been realized by many researchers [67, 68] that for automatic speech recognition/understanding systems to function efficiently, then high level linguistic processing is a must. Examples of linguistic knowledge source modules exploited in the speech recognition process are the syntactic and the semantic modules [69]. Inclusion of such knowledge source modules implies that we are able to evaluate possible word hypotheses as to the linguistic composition of the utterance spoken. The arrangements of these knowledge source modules in the general structure of the speech recognition system makes it possible to perform sequential or parallel processing. Both processing procedures (i.e., sequential or parallel) have their own advantages and disadvantages.

A possible structure for a knowledge based strategy for an Arabic speech recognition system is shown in Figure 54. The first stage of the

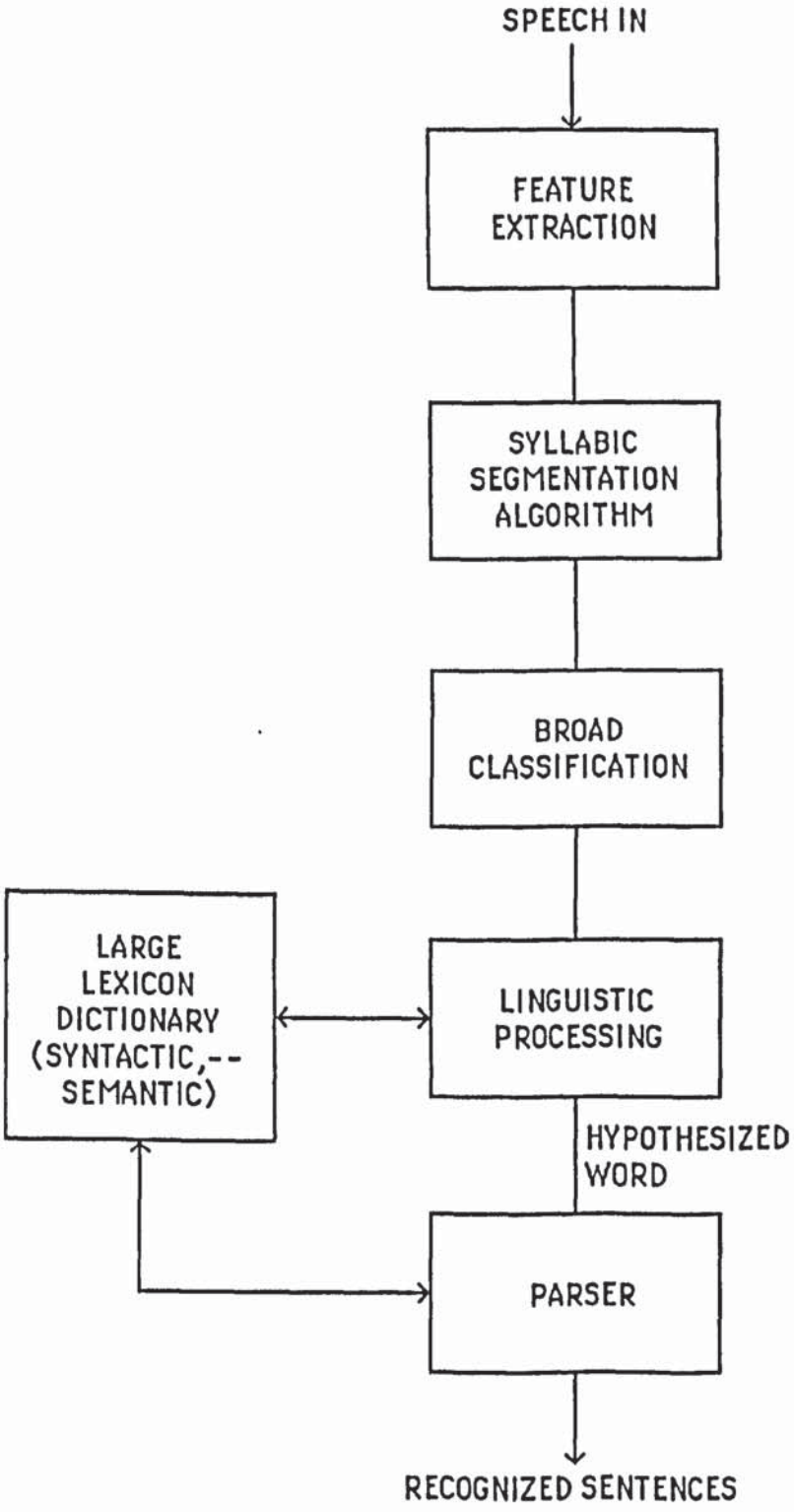


FIGURE 54

STRUCTURE OF KNOWLEDGE BASED ARABIC SPEECH RECOGNITION SYSTEM.

recognition process is the acoustic-phonetic module. The output of this module is information which will enable broad phonetic classification to take place.

The Arabic syllabic segmentation algorithm is utilized in the implementation of this module.

7.4.2 Aim of the Arabic Acoustic-Phonetic (Syllabic) Module.

The main objective of the Arabic Acoustic-phonetic (syllabic) module is to characterize the Arabic spoken utterance in terms of the following broad phonetic knowledge:

1. Syllabic count.
2. Stressed syllable identification.
3. Syllables types, i.e., short, medium or long.
4. Formant values of vowel regions.
5. Voicing degree of consonant clusters.
6. Presence of Gemination.

This characterization of syllable units will reduce the number of hypothesized words candidates stored in a large lexicon dictionary.

7.4.3 Broad Phonetic Classification.

The output of the Arabic syllabic segmentation algorithm is a set of maxima and minima points representing the syllabic structure of the spoken utterance. These points are processed to determine the following:

1. ***Syllabic count.*** This is merely the output of the Arabic syllabic segmentation algorithm.
2. ***Syllabic stress identification.*** The highest energy maximum point generally corresponds with the stressed syllable.
3. ***Syllables types.*** Syllables durations (distance between minima points x 6.4 mseo.) are normalized with respeot to whole utterance duration. Classification into syllable types is achieved by examining the length of durations.
4. ***Formant values of vowels segments.*** Prominant maxima points of the energy oontour generally correspond to vowels segments. Hence, linear prediotion analysis is applied to those speech segments corresponding to maxima points. The smoothed power speotrum of Arabic vowels using LP representation is obtained using the following formula (35) given below:

$$P(\Omega) = \frac{G^2}{\rho(o) + 2\sum_{i=1}^{P} \rho(i) \cos(i\Omega)} \qquad (35)$$

where,

$\Omega = 2\pi fT$,

T = Sampling frequenoy,

$$G = R(o) + \sum_{k=1}^{p} a_k R(k), \qquad (36)$$

and

$$\rho(i) = \sum_{k=o}^{p-i} a_k a_{k+1}; \quad a_o = 1, \quad 1 \leqslant i \leqslant p \qquad (37)$$

G is the gain, $a_1 \ldots a_p$ are the LP coefficients, $R_1 \ldots R_p$ are the autocorrelation coefficients. $P(\Omega)$ is computed at equidistant 250 points in the range $0 \leqslant \Omega \leqslant \pi$, corresponding to frequency range of $0 \leqslant f \leqslant 5$ *kHz*. The estimated formant values are obtained by a peak-picking algorithm.

5. ***Voicing degree of consonant clusters.*** Minima points corresponding to speech segments of consonants are classified into silence, voiced and voiceless categories according to the energy values of these minima points.

6. ***Presence of Gemination.*** Gemination is detected by the presence of large width minimum point in the energy contour. The width is measured as the distance between the two points on either side off the minimum point. These two points are taken to be the ones which are 2% higher in energy value than the minimum point energy value.

7.4.4 *Arabic Acoustic-Phonetic Module Software System Implementation.*

Figure 55 shows a flowchart of the Arabic acoustic-phonetic (syllabic) module. The module is realized by the series of procedures given in Appendix G using the developed PC-based speech processing system.

7.4.5 *Preliminary Classification Results and Discussions.*

Random Arabic words were initially used to tune-in the threshold detection levels for syllable types, voicing and Gemination. Random Arabic words were then applied, not to fully evaluate the performance of the Arabic acoustic-phonetic module by giving percentages about success rates of classification, but rather to high-light the importance of this module and to shed light on the main analytical components of the module. Successful adaptive threshold detection levels for classifying the degree of voicing for consonants and the presence of Gemination are shown in Tables 25 and 26, respectively. Syllable type classifications by simple adaptive threshold level techniques were not found to be highly successful. However, occasionally correct classifications were obtained when using the threshold levels shown in Table 27.

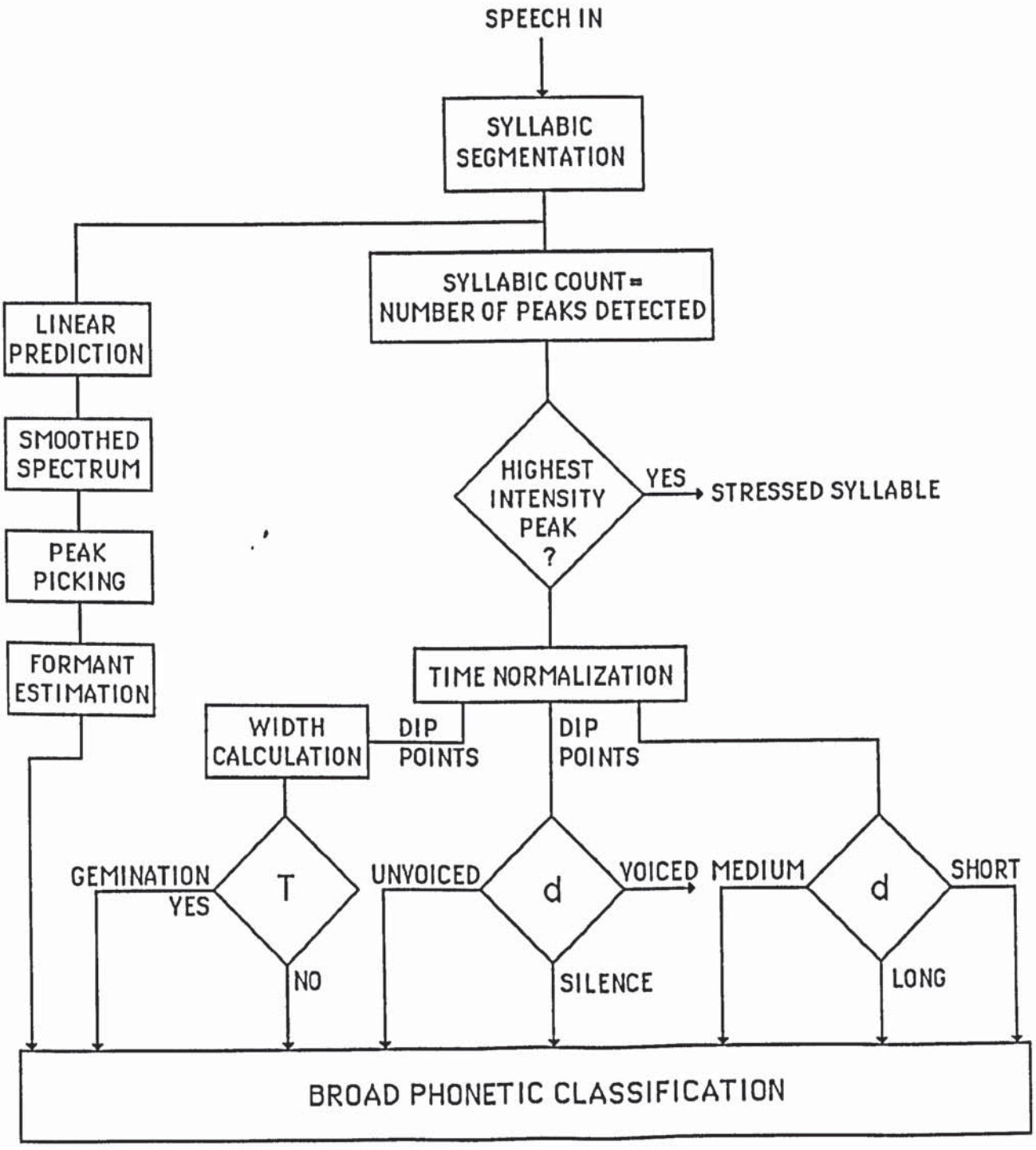


FIGURE 55

FLOWCHART OF THE ARABIC ACOUSTIC-PHONETIC MODULE

Threshold level d Maximum Intensity=1	Consonant Classification
$d \geq 0.1$	voiced
$0.02 \leq d \leq 0.05$	voiceless
$d \leq 0.02$	silence

Table 25
Threshold levels for Voiced/Unvoiced Classification of Consonants.

Width value T Maximum Duration=1.0	Gemination
$T > 0.15$	present
$T < 0.15$	not present

Table 26
Threshold levels for Detection of Geminated Consonants.

Threshold level d Maximum Duration=1	Syllable Types Classification
$d < 0.35$	short
$0.35 \leq d \leq 0.55$	medium
$d \geq 0.55$	long

Table 27
Threshold levels for Syllable Types Classification.

Formant values of vowel regions are estimated for the smoothed spectrum by peak picking. This method is prone to omission errors (two formants close together are merged in one peak). Peaks in the spectrum generally correspond to formants and voice harmonics. The maximum number of peaks do not exceed five. The rules for rejecting peaks due to voice harmonics are as follows:

1. If first peak is < 250 Hz, then the second peak is taken as F1, third peak as F2 and so on.
2. If first peak is > 800 Hz, then formant estimation is terminated.

The following is the output of the acoustic-phonetic module when the Arabic utterance /juna:di/ was entered. It is made of three syllables, the first is of the short type and the rest are medium. All consonant clusters are voiced.

1. Syllabic count = 3.
2. Stressed syllable = Syllable number 2.
3. Syllables types are low, medium and high.
4. First vowel F1, F2 and F3 values are 340 Hz, 1320 Hz and 2560 Hz, respectively.
5. Second vowel F1, F2 and F3 values are 240 Hz, 1500 Hz and 2760 Hz, respectively.
6. Third vowel F1, F2 and F3 values are 680 Hz, 1560 Hz and 3000 Hz, respectively.

7. First consonant cluster is voiced.

7.4.6 Conclusion.

The main purpose behind this application is to show the usefulness and applicability of the Arabic syllabic segmentation algorithm in the implementation of an acoustic-phonetic (syllabic) module which can be an essential component for a knowledge based strategy for Arabic speech recognition.

CHAPTER 8

Chapter 8

OVERALL CONCLUSION

8.1 *Overall Conclusion*

This thesis reports the findings on the subject of Arabic Speech Processing with particular emphasis on Arabic speech recognition. The following are the main points of the investigations carried out:

Arabic Phonetic System. A thorough review of Arabic phonology was reported. The syllabic nature of the Arabic language was highlighted in terms of syllable types, syllable structures and primary stress rules. This has implied that a computational algorithm can be developed to decompose Arabic speech into syllabic units.

Literature Survey. A complete literature survey was carried out. Speech analysis, synthesis and recognition techniques were reviewed. Arabic speech synthesis and recognition activities have been also reviewed. It is noted that most research activities have been towards Arabic speech synthesis and the amount of published research work in Arabic speech processing is negligible as compared to other languages. Hence, this thesis can be looked at as a step towards promoting research activities in Arabic speech processing.

PC-based Speech Processing System. To facilitate research in speech processing, tools need to be developed or purchased, which can be very expensive. Hence, to gain control and flexibility, a PC-based speech processing system was developed. The system is capable of digitizing, storing and analyzing speech samples. The system was utilized in the development of the Arabic syllabic segmentation algorithm, and in various aspects of Arabic speech recognition applications.

Acoustical Properties of Arabic Similar Sounds. Spectrographic analyses were performed on two similar Arabic sounds, namely, (/θ/, /f/) and (/d/, /ḍ/). The purpose was to determine salient features that fully characterize each phoneme.

For the Arabic consonant pair (/θ/, /f/), it was shown that the transitional behavior of F1 in the context of different vowels can be looked at as the identification cue.

For the Arabic consonant pair (/d/, /ḍ/), it was shown that a clear appreciable salient feature is not evident except from the point of determining burst formant values and the influence d makes on F2 values of neighboring vowels.

Syllabic Segmentation. An automatic Arabic syllabic segmentation algorithm was developed, which is based on short time energy computation. The

syllabic segmentation algorithm was tested using a variety of monosyllabic and multisyllabic words, and its overall performance was found to be satisfactory. The main parameters affecting the accuracy of segmentation were the adaptive threshold values. The main advantage of the algorithm is that it is simple and easy to realize on a computer program.

Arabic Speech Recognition Applications. Three important aspects of Arabic . speech recognition applicatoins were demonstrated. All three applications make use of the developed syllabic segmentation algorithm.

An automatic Arabic vowel recognition system was implemented. Different parametric representations of the steady state Arabic vowel segments were studied and the performance of each parametric representation was measured using different distance measure techniques.

An isolated Arabic word recognition system was implemented using stressed syllabic units as the primary units for speech recognition. The system was tested with Arabic digits. Good performance results were achieved for speaker-dependent modes.

An Arabic acoustic-phonetic (syllabic) module was implemented. This module is one component of a knowledge-based strategy for Arabic speech recognition. The module is able to transcribe

utterances in a broad-phonetic sense. Broad phonetic classification reduces word candidates, hence achieving higher recognition processing rates and larger lexicon vocabulary.

8.2 Future Work

This thesis is considered to be a step forward towards creating and sustaining research work in Arabic speech processing which complements the Arabization efforts of the Kuwait Institute for Scientific Research in developing appropriate tools (morphology and syntax) for the processing of the Arabic language. The main hardware and software facilities required to perform research in speech processing have already been developed. A few suggestions for future work, in line of what has already been accomplished so far, include the following:

1. Detailed study of the acoustic-phonetic characteristics of the Arabic language.
2. Study of the prosodic features of the Arabic language and their use for aiding recognition.
3. Experimenting with larger vocabularies in syllabic based units for isolated Arabic word speech systems.
4. Extending the Arabic syllabic segmentation algorithm to connected speech.

5. Developing isolated Arabic word recognition systems for Arabic word spotting and speaker verification.

6. Developing and implementing a continuous Arabic speech recognition system employing a language knowledge-based approach for speech recognition.

APPENDICES

Appendix A

Sample Arabic Text

Appendix B

MC145414 Dual Tunable Low-Pass Sampled Data Filters

Appendix C

Speech Processing Routines

```
program speech;
label 1000,2000;
const
  N:integer=256;
type
 ss=string[30];
var
  hhh:array[1..8]of ss;
  data,zcr,                {signal array & quantized.array)  -
  qdata                    : array [0..8191] of integer;
  sum                      : array [0..130] of integer;
min,lmax,lmin,spread,lpoint,rpoint,                  { array hold
  emax                     : array [0..24] of integer;
  sframe,                  {end point detection)
  eframe,
  dur,                     {duration and actual duration)
  adur,
  max,                     {max of signal & max of sum)
  maxx,zmax,
  no_peaks,nof_dips,nof_peaks,nhi,g,i,j              :integer;
  ch:char;
{$I GRAPH.P)
 {**procedure to read data from file specified to array data
    procedure read_data;
    type
    rec_type    = record
                    low,
                    high : char
                  end;

   var
   rec          : rec_type;
   code_file    : file of rec_type;
   name         : string[20];
   i            : integer;
   begin

    write('Enter file name : ');
    readln(name);
    graphmode;textmode;
    gotoxy(20,10);write('READING FILE ',name);
    assign(code_file,name);
    reset(code_file);
    seek(code_file,256);
    i:=0;max:=0;
    repeat
     read(code_file,rec);
     with rec do
      begin
        data[i]:=ord(high)*256+ord(low)-2048;
        if abs(data[i]) >= max then max:=abs(data[i]);
      end;
```

```
    i:=i+1;
   until eof(code_file);
 for i:=0 to 8191 do qdata[i]:=round(100*(data[i] / max));
  end;


{**procedure to display waveform plus curser to pick end poi
  procedure wave;

  var
  num,
  i,j,k,
  x1,
  y1    : integer;
  cur   : char;
  lin,
  pic   : array [0..2101 of char;

  begin
   graphmode;
   x1:=0;
   y1:=50;
   for i:=0 to 8191 do
    begin
     num:=qdata[i] div 2 ;
     if num>=0 then num:=50-num
     else num:=-num+50;
     draw(x1,y1,round(i/8191*300),num,1);
     x1:=round(i/8191*300);
     y1:=num;
    end;
   getpic(pic,0,0,0,100);
   draw(0,0,0,100,2);
   getpic(lin,0,0,0,100);
   i:=0;
   repeat
    read(Kbd,cur);
    case cur of
     'f' : begin
             i:=i+1;
             putpic(pic,round((i-1)/8191*300),100);
           getpic(pic,round(i/8191*300),0,round(i/8191*300),
             putpic(lin,round(i/8191*300),100);
             gotoxy(35,2);
             writeln(i);
           end;
     'b' : begin
             i:=i-1;
             putpic(pic,round((i+1)/8191*300),100);
           getpic(pic,round(i/8191*300),0,round(i/8191*300),
             putpic(lin,round(i/8191*300),100);
```

```
                gotoxy(35,2);
                writeln(i);
              end;
      'F' : begin
                i:=i+50;
                putpic(pic,round((i-50)/8191*300),100);
             getpic(pic,round(i/8191*300),0,round(i/8191*300),
                putpic(lin,round(i/8191*300),100);
                gotoxy(35,2);
                writeln(i);
              end;
       'p' : begin
                x1:=0;
                y1:=150;
                j:=0;
                for k:=i to i+255 do
                 begin
                  num:=qdata[k] div 2 ;
                  if num>=0 then num:=150-num
                  else num:=-num+150;
                  draw(x1,y1,round(j/255*300),num,1);
                  x1:=round(j/255*300);
                  y1:=num;
                  j:=j+1;
                 end;
               end;
       'w' : begin
                x1:=0;
                y1:=150;
                j:=0;
                for k:=i to i+255 do
                 begin
                  num:=qdata[k] div 2 ;
                  if num>=0 then num:=150-num
                  else num:=-num+150;
                  draw(x1,y1,round(j/255*300),num,0);
                  x1:=round(j/255*300);
                  y1:=num;
                  j:=j+1;
                 end;
               end;

       ' ' : sframe:=i;
       't' : eframe:=i;
    end;
  until cur='t';
 dur:=eframe-sframe;
end;


procedure hide;
```

```
type
regtype = record
          ax,bx,cx,dx,bp,si,di,ds,es,flags : integer;
        end;


var
regs : regtype;
begin
 regs.ax:=256;
 regs.cx:=16*256;
 intr(16,regs);
end;

procedure enable;

type
regtype = record
          ax,bx,cx,dx,bp,si,di,ds,es,flags : integer;
         end;


var
regs : regtype;
begin
 regs.ax:=256;
 regs.cx:=8*256+8;
 intr(16,regs);
end;


procedure energy;

var
  i,k,num,
  j,
  n,
  x1,
  z,
  y1     : integer;
  test   : boolean;

begin
  j:=0;n:=0;k:=0;z:=0;
  textmode;
  for i:=0 to 130 do sum[i]:=0;
  for i:=0 to 130 do zcr[i]:=0;
  test:=qdata[0]>0;
  repeat
    for i:=k to k+255 do
      begin
        sum[j]:=sum[j]+abs(qdata[i]);
```

```
        n:=n+1;
        if (test) xor (qdata[i]<0) then
           else
             begin
               z:=z+1;
               test:=qdata[i]>0;
             end;
      end;
    k:=k+64;
    n:=0;
    {writeln(j,' ',sum[j]);}
    zcr[j]:=z;{***writeln(j,' ',zcr[j]);***}
    z:=0;
    j:=j+1;
  until k + 255 > 8191 ;
  maxx:=0;zmax:=0;
  adur:= j-1;
  for i:=0 to j-1 do if sum[i]>maxx then maxx:=sum[i] ;
  for i:=0 to j-1 do if zcr[i]>zmax then zmax:=zcr[i] ;
end;
procedure zero ing;

var
  i,k,
  j,num,
  n,
  x1,
  y1     : integer;

begin
 graphmode;
 j:=adur+1;
 x1:=70;y1:=150;
    for i:=0 to j-1 do
    begin
   draw(x1,y1,70+round(i/(j-1)*200),150-round(zcr[i]/zmax*15
      x1:=70+round(i/(j-1)*200);
      y1:=150-round(zcr[i]/zmax*150);
    end;
  draw(70,150,270,150,3);
  draw(70,150,70,0,3);
  draw(70,0,270,0,3);
  draw(270,0,270,150,3);
  {***routine to plot data***}
  x1:=70;
  y1:=175;
  for i:=0 to 8191 do
    begin
      num:=qdata[i] div 4;
      if num>=0 then num:=175-num
      else num:=-num+175;
      draw(x1,y1,70+round(i/(8191)*200),num,1);
```

```
      x1:=70+round(i/(8191)*200);
      y1:=num;
    end;
  gotoxy(1,5);write('Zero');
  gotoxy(1,7);write('Crossing');
end;


procedure plot_energy;

var
  i,k,
  j,num,
  n,
  x1,
  y1     : integer;

begin
 graphmode;
 j:=adur+1;
 x1:=70;y1:=150;
    for i:=0 to j-1 do
    begin
   draw(x1,y1,70+round(i/(j-1)*200),150-round(sum[i]/maxx*15
      x1:=70+round(i/(j-1)*200);
      y1:=150-round(sum[i]/maxx*150);
    end;
  draw(70,150,270,150,3);
  draw(70,150,70,0,3);
  draw(70,0,270,0,3);
  draw(270,0,270,150,3);
  {***routine to plot data***}
  x1:=70;
  y1:=175;
  for i:=0 to 8191 do
    begin
      num:=qdata[i] div 4;
      if num>=0 then num:=175-num
      else num:=-num+175;
      draw(x1,y1,70+round(i/(8191)*200),num,1);
      x1:=70+round(i/(8191)*200);
      y1:=num;
    end;
   gotoxy(1,5);write('Absolute');
   gotoxy(1,7);write('Short');
   gotoxy(1,9);write('Time');
   gotoxy(1,11);write('Energy');
   gotoxy(30,20);write('Time');
 end;

   procedure fft;
   const
```

```
n:integer=256;
pi:real=3.1415926;
r:real=2.3025851;
var
i,j,k,n1,n2,i1,l,l1,l2,x1,y1,rep          :integer;
t1,t2,u1,u2,u3,u4,v1,v2,w1,w2               :real;
a:array[1..256,0..1]of real;
 begin
     {read data in a_array}
     for j:=0 to 1 do
      begin
       for i:=1 to n do a[i,j]:=0;
      end;
  for i:=1 to n do a[i,0]:=(data[i+sframe]/max)*(0.54-0.46*
    {****bit revesal****}
    n2:=n div 2 ;
    n1:=n-1;
    j:=1;
    for i:=1 to n1 do
        begin
          if j>i then
            begin
              t1:=a[j,0];
              t2:=a[j,1];
              a[j,0]:=a[i,0];
              a[j,1]:=a[i,1];
              a[i,0]:=t1;
              a[i,1]:=t2;
            end;
          k:=n2;
          while j>k do
            begin
              j:=j-k;
              k:=round(k/2);
            end;
         j:=j+k;
        end;
     {***end of bit reversal***}
     {***                    ***}
     for l:=1 to 8 do
       begin
          l1:=round(exp(l*ln(2)));
          l2:=round(l1 / 2);
          u1:=1;
          u2:=0;
          w1:=cos(pi / l2);
          w2:=-1*sin(pi / l2);
          for j:=1 to l2 do
           begin
             i:=j;
            repeat
             i1:=i+l2;
```

```
                v1:=(a[i1,0]*u1-a[i1,1]*u2);
                v2:=(a[i1,1]*u1+a[i1,0]*u2);
                a[i1,0]:=a[i,0]-v1;
                a[i1,1]:=a[i,1]-v2;
                a[i,0]:=a[i,0]+v1;
                a[i,1]:=a[i,1]+v2;
                i:=i+i1;
              until i>n;
             {***     ***}
              u3:=u1;
              u4:=u2;
              u1:=(u3*w1-u4*w2);
              u2:=(u4*w1+u3*w2);
             end;
            end;
            graphmode;
            x1:=50;y1:=100;
            for i:=1 to n div 2 do
             begin
               t1:=sqrt(sqr(a[i,0])+sqr(a[i,1]));
            draw(x1,y1,50+round(i/128*128),80-round(20*ln(t1)/r
               x1:=50+round(i/128*128);
               y1:=80-round(20*ln(t1)/r);
             end;
            gotoxy(14,4);write('FFT');draw(150,27,160,27,1);
      end;


procedure lpc;
const
p:integer=14;
pi:real=3.1415926;
var
  i,
  j,rep,maa_x          : integer;
  be,ww,ph,v          :real;
  wdata,spec              : array [0..255] of real;
  r,row,
  ki,e                :array [0..15] of real;
  alfa                :array [0..15,0..15] of real;
  lmax,lmin,form          :array [0..8] of integer;
begin
  {programe to calculate lpc coef 15)
or i:=0 to N-1 do wdata[i]:=data[i+sframe]/max*(0.54-0.46*co
  {now calculate autocorraltions coefficents in R[0..p])

    for i:=0 to p do
    begin
     R[i]:=0;
      for j:=0 to N-1-i do R[i]:=R[i]+wdata[j]*wdata[j+i]
    end;
```

```
  {*** now write to file***}
{recursive procedure Durbins for lp}
  be:=0;
  E[0]:=R[0];
  for i:=1 to 14 do
  begin
     for j:=1 to i-1 do
     begin
      be:=be+alfa[(i-1),j]*R[i-j];
     end;
     K[i]:=-(R[i]+be)/E[i-1];
     alfa[i,i]:=K[i];
      for j:=1 to i-1 do
       begin
        alfa[i,j]:=alfa[i-1,j]+K[i]*alfa[(i-1),(i-j)];
       end;
     E[i]:=(1-sqr(K[i]))*E[i-1];
     be:=0;
   end;
   {cal Gain V}
   V:=0;
   for i:=1 to 14 do V:=V+alfa[14,i]*R[i];
   V:=V+R[0];
  {for i:=1 to 14 do writeln(alfa[14,i]);}
  {now plot spectrum}
   for i:=0 to 14 do row[i]:=0;
   alfa[14,0]:=1;
   for i:=0 to 14 do
    begin
     for j:=0 to 14-i do
      begin
       row[i]:=row[i]+alfa[14,j]*alfa[14,j+i];
      end;
    end;
  {spectrum}
   ww:=0;ph:=0;
   j:=0;
   repeat
    for i:=1 to 14 do
      begin
       ww:=ww+row[i]*cos(i*ph*pi/2500);
      end;
    ph:=ph+19.53;
    spec[j]:=ln(sqr(V)/(row[0]+2*ww));
    j:=j+1;
    ww:=0;
   until j>127;
   (**graphmode;**)
   for i:=0 to 127 do plot(i+50,100-round(10*spec[i]),3);
   gotoxy(14,5);write('LPC');draw(150,35,160,35,3);
end;
```

```
procedure frame;
var
 i,j,k,x1,y1,num:integer;
  begin
     (**draw(50,50,178,50,3);**)
     draw(50,50,50,150,3);
     (**draw(50,150,178,150,3);**)
     draw(178,50,178,150,3);
     x1:=50;
     y1:=175;
    j:=0;
     for k:=0 to 255 do
              begin
               num:=qdata[k+sframe] div 4 ;
               if num>=0 then num:=175-num
               else num:=-num+175;
               draw(x1,y1,50+round(j/255*128),num,1);
               x1:=50+round(j/255*128);
               y1:=num;
               j:=j+1;
              end;

     end;

 procedure ifft;
 label 10;
 const
 n:integer=256;
 pi:real=3.1415926;
 r:real=2.3025851;
 var
,j,k,n1,n2,i1,l,l1,l2,s1,ku,x1,y1,rep                    :integer
 t1,t2,u1,u2,u3,u4,v1,v2,w1,w2                  :real;
 a:array[1..256,0..1]of real;
  begin
     {read data in a_array}
     for j:=0 to 1 do
      begin
       for i:=1 to n do a[i,j]:=0;
      end;
  for i:=1 to n do a[i,0]:=(data[i+sframe]/max)*(0.54-0.46*c
     {for i:=1 to n do writeln(i,' ',a[i,0]);}
     {finsih}
     s1:=-1;ku:=0;
     {for i:=1 to 10 do writeln(a[i,0]);}
     {****bit revesal****}
10:  n2:=n div 2 ;
     n1:=n-1;
     j:=1;
     for i:=1 to n1 do
         begin
           if j>i then
```

```
        begin
          t1:=a[j,0];
          t2:=a[j,1];
          a[j,0]:=a[i,0];
          a[j,1]:=a[i,1];
          a[i,0]:=t1;
          a[i,1]:=t2;
        end;
      k:=n2;
      while j>k do
        begin
          j:=j-k;
          k:=round(k/2);
        end;
     j:=j+k;
    end;
{***end of bit reversal***}
{for i:=1 to n do writeln(i,' ',a[i,0]);}
{***                    ***}
for l:=1 to 8 do
  begin
    l1:=round(exp(l*ln(2)));
    l2:=round(l1 / 2);
    u1:=1;
    u2:=0;
    w1:=cos(pi / l2);
    w2:=s1*sin(pi / l2);
    for j:=1 to l2 do
     begin
       i:=j;
      repeat
       i1:=i+l2;
       v1:=(a[i1,0]*u1-a[i1,1]*u2);
       v2:=(a[i1,1]*u1+a[i1,0]*u2);
       a[i1,0]:=a[i,0]-v1;
       a[i1,1]:=a[i,1]-v2;
       a[i,0]:=a[i,0]+v1;
       a[i,1]:=a[i,1]+v2;
       i:=i+l1;
      until i>n;
     {***    ***}
      u3:=u1;
      u4:=u2;
      u1:=(u3*w1-u4*w2);
      u2:=(u4*w1+u3*w2);
     end;
    end;
   { graphmode;}
    for i:=1 to n do
    begin
      t1:=sqrt(sqr(a[i,0])+sqr(a[i,1]));
      a[i,0]:=(20*ln(t1)/r);
```

```
            a[i,1]:=0;
          end;
          s1:=1;
          ku:=ku+1;
          if ku=1 then goto 10;
          graphmode;
          x1:=50;y1:=100;
          for i:=1 to n div 2 do
          begin
         draw(x1,y1,50+round(i/128*128),100-round(a[i,0]),1)
            x1:=50+round(i/128*128);
            y1:=100-round(a[i,0]);
          end;
          (*find pitch*)
          t1:=0;for i:=40 to n div 2 do if a[i,0]>t1 then
                                                  begin
                                                 t1:=a[i,0
                                                    j:=i;
                                                  end;
          gotoxy(12,4);write('Cepstrum');
          gotoxy(12,5);write('Pitch = ',10000 div j);

    end;
```

```
(**main program**)
begin
  textbackground(0);
lrscr;window(15,8,60,18);textcolor(10);textbackground(12);cl
  sound(300);delay(100);nosound;
  gotoxy(14,2);write('SPEECH ANALYSIS PROGRAM');
  gotoxy(23,6);write('By');
  gotoxy(15,7);write('Abdulhadi Al-Otaibi');
  GOTOXY(16,8);WRITE('Aston University');
otoxy(4,9);write('Kuwait Institute For Scientific Research')
  gotoxy(10,10);write('Kuwait PO Box 24885 Safat');
eadln;textbackground(0);clrscr;window(30,10,80,10);read data
0: window(1,1,80,25);textbackground(0);clrscr;textcolor(24);
     gotoxy(30,8); write('SELECT ONE');
     textcolor(2);textbackground(9);
      hhh[1]:='  DISPLAY WAVEFORM          ';
      hhh[2]:='  COMPUTE & DISPLAY ENERGY';
      hhh[3]:='  DISPLAY ZERO_CROSSING     ';
      hhh[4]:='  FFT & LPC                 ';
      hhh[5]:='  CEPSTRUM PROCESSING       ';
      hhh[6]:='  EXIT TO DOS               ';
      for nh1:=1 to 6 do
               begin
```

```
                gotoxy(30,10+nh1);
                write(hhh[nh1]);
              end;
 textbackground(12);gotoxy(30,11);write(hhh[1]);g:=1;
repeat
 hide;read(Kbd,ch);enable;
    if (ch='p') then
                begin
                  if g=87 then
                           begin
                            textbackground(9);
                            gotoxy(30,10+g);
                            write(hhh[g]);
                            g:=0;
                           end;

                   textbackground(9);
                   gotoxy(30,10+g);
                   write(hhh[g]);
                   if g=6 then g:=0;
                   textbackground(12);
                   g:=g+1;
                   gotoxy(30,10+g);
                   write(hhh[g]);

                  end;
         until ch='k';
 if g=1 then
             begin
              wave;
              readln;
             end;
 if g=2 then
             begin
             energy;
             plot_energy;
             readln;
             end;
 if g=3 then
             begin
              zero_ing;
              readln;
             end;

 if g=4 then
             begin
             fft;
             lpc;
             frame;
             readln;
             end;
 if g=5 then
```

```
                    begin
                    ifft;
                    frame;
                    readln;
                    end;
        if g=6 then goto 2000;
       goto 1000;
2000:
end.
```

Appendix D

Arabic Syllabic Segmentation Algorithm

Software Routines

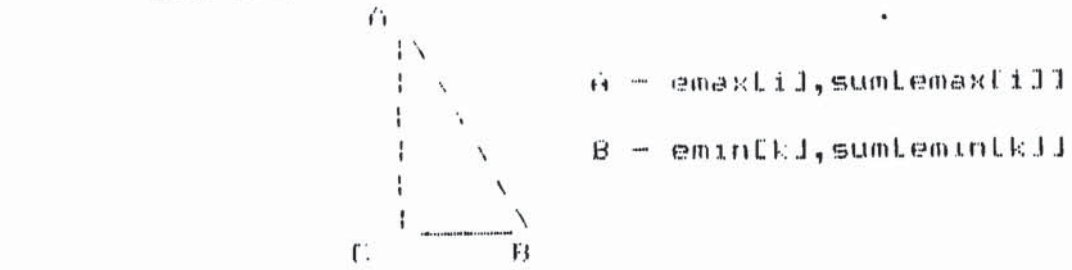
```
(* procedure to compute energy *)
procedure energy;

var
   i,k,num,
   j,
   x1,
   z,
   y1   . : integer;
   test   : boolean;

begin
   j:=0;k:=0;z:=0;
   textmode;
   for i:=0 to 130 do sum[i]:=0;
   for i:=0 to 130 do zcr[i]:=0;
   test:=qdata[0]>0;
   repeat
      for i:=k to k+255 do
        begin
          sum[j]:=sum[j]+abs(qdata[i]);

          if (test) xor (qdata[i]<0) then
             else
              . begin
                  z:=z+1;
                  test:=qdata[i]>0;
                end;
        end;
      k:=k+64;

      zcr[j]:=z;
      z:=0;
      j:=j+1;
   until k + 255 > dur ;
   maxx:=0;zmax:=0;
   x1:=60;
   y1:=150;
   adur:= j-1;
   for i:=0 to j-1 do if sum[i]>maxx then maxx:=sum[i] ;
   for i:=0 to j-1 do if zcr[i]>zmax then zmax:=zcr[i] ;
end;


(* low pass filter*)
procedure filter;
const pi:real=3.1415927;
var
```

```
  fdata:array[-2..8000]of real;
  num,
  j,
  noise,
  i,
  x1,
  y1,m      : integer;
  k,t,
  b1,
  b2        : real;

begin
  (**writeln( bw');read(x1);**)
  m:=eframe-sframe;
  for i:=-2 to m do fdata[i]:=0;


  t:=0;
  for i:=0 to m do if abs(fdata[i])>t then t:=abs(fdata[i]);
  for i:=0 to m do qdata[i]:=round(fdata[i]/t*100);
  (**
  graphmode;
  x1:=0;
  y1:=100;
  for i:=0 to m do
    begin
      num:=qdata[i];
      if num>=0 then num:=100-num
      else num:=-num+100;
      draw(x1,y1,round(i/m*300),num,1);
      x1:=round(i/m*300);
      y1:=num;
    end;
    readln;
    **)
 end;


procedure peak_detection;

var
  i,k,
  j          : integer;
begin
  for i:=0 to 24 do emax[i]:=-1;
  for i:=0 to 24 do emin[i]:=-1;
  i := 1;
  j := 0;
```

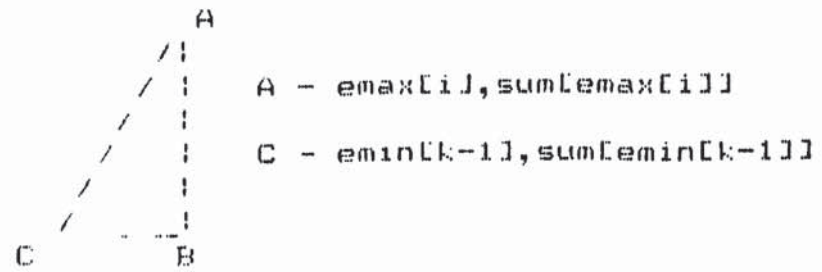
```
(*
  This repeat loop picks up the 50% transition points
*)
  repeat
(*

*)
    while (i<adur) and (emax[j]<0) do
      begin
        if sum[i] <= sum[i+1] then i:=i+1
        else emax[j]:=i;
      end;
    if (i=adur) and (emax[j]=-1) then emax[j] := adur;
    i := i + 1;
(*

*)
    while (i<adur) and (emin[j]<0) do
      begin
        if sum[i] >= sum[i+1] then i:=i+1
        else emin[j]:=i;
      end;
    if (i=adur) and (emax[j]<>-1) then emin[j] := adur;
    i := i + 1;
    j := j+1;
  until i>=adur;
  {***writeln;
  writeln(adur);
  write('Max : ');
  for i:= 0 to j-1 do write(emax[i]:4);
  writeln;
  for i:= 0 to j-1 do write(emin[i]:4);***)
  no_peaks:=j-1;
end;


 procedure syl_rec;
 var
  i,j,k,i1,l1,j1,
  difx,dify    : integer;
  contr        : boolean;

 (*

   the following stages

   (i) maxima of the max and minima of the min principle
       (process_extrema)
```

```
  (ii) difx and dify
  (iii)minima more than 70% of maxx

  (v)  maxima less than 20% of maxx


*)

 begin
  textmode;
  i:=0;
  k:=0;
  j:=no_peaks+1;
  for i:=i-1 downto 0 do emin[i+1]:=emin[i];
  emin[0]:=0;
  nof_peaks:=j;
  nof_dips:=j+1;
  (*

    dif-y - 10% of the span on the y-axis which is sum
  *)
  difx:=round(0.04*adur);
  dify:=round(0.1*maxx);
  sum[0]:=0;

  writeln;writeln(adur:5,difx:5,maxx:5,dify:5);writeln;
  write('Max : ');writeln;
  for i:= 0 to nof_peaks-1 do write(emax[i]:4);
  writeln;writeln;
  for i:=0 to nof_dips-1 do write(emin[i]:4);
  writeln;
  writeln;

  (*

  *)
  j:=nof_peaks;
  i:=0;
(*  while i<nof_peaks do
    begin
      write(sum[emin[i]]:5);
      write(sum[emax[i]]:5);
      i:=i+1;
      if (i mod 3)=0 then writeln;
    end;
  writeln;
*)
```

```
i:=0;
k:=1;
repeat
  (* Down slope
```




```
    The next if checks if
                              AB < dify
                              and
                              BC < difx
  *)


    begin
      (*

        the one with a higher value (sum)
      *)
      nof_peaks:=nof_peaks-1;
      (* compress emax *)
      for i1:=1 to nof_peaks-1 do emax[i1]:=emax[i1+1];
      nof_dips:=nof_dips-1;
      (* set up the correct emin to reject *)
      if sumlemin[k]]>sum[emin[k-1]] then j1:=k
      else j1:=k-1;
      (* compress emin *)
      for i1:=i1 to nof_dips-1 do emin[i1]:=emin[i1+1];
    end
  else
  (*  Up slope
```




```
    The next if checks if
                              AB < dify
```

```
                                    and
                                    BC < difx
    *)

        begin
        (*

          the one with a higher value (sum)
        *)
        nof_peaks:=nof_peaks-1;
        (* compress emax *)
          for j1:=i to nof_peaks-1 do emax[j1]:=emax[j1+1];
          nof_dips:=nof_dips-1;
          if sum[emin[k]]>sum[emin[k-1]] then j1:=k
          else j1:=k-1;
        (* compress emin *)
          for j1:=j1 to nof_dips-1 do emin[j1]:=emin[j1+1];
        end
      else
        begin
          (*
            Retains emax[i] and emin[k]
          *)
          i:=i+1;
          k:=k+1;
        end;
    until (i=nof_peaks);

    write('Difx/Difv');
    readln;
    write('Max : ');writeln;
    for i:= 0 to nof_peaks-1 do write(emax[i]:4);
    writeln;writeln;
    for i:=0 to nof_dips-1 do write(emin[i]:4);
    writeln;

  (* rejects minima more than 70% of maxx *)
    i:=0;
    k:=0;
    repeat
      (*

        of the maximum
      *)
      if sum[emin[k]]<(0.7*maxx) then
        begin
          emin[i]:=emin[k];
          i:=i+1;
```

```
      end;
    k:=k+1;
  until k=nof_dips;
  nof_dips:=i;

  write('Minima more than 70%');
  readln;
  write('Max :  ');writeln;
  for i:= 0 to nof_peaks-1 do write(emax[i]:4);
  writeln;writeln;
  for i:=0 to nof_dips-1 do write(emin[i]:4);
  writeln;


(*

   in the case that between some two consecutive minima
   there are now more than one maxima

   any two minima by rejecting all but the largest of the
   maxima for a case stated above
*)
(* Picks max of maxima between two min *)
  i:=0;
  i1:=0;
  l1:=0;
  repeat
    while (i<nof_peaks) and (emax[i]<emin[i1]) do i:=i+1;
    i1:=i1+1;
    if (i<nof_peaks) and (emax[i]<emin[i1]) then
      begin
        lmax[l1]:=emax[i];
        while (emax[i]<emin[i1]) and (i<nof_peaks) do
          begin

            i:=i+1;
          end;
        l1:=l1+1;
      end;
  until (i1=nof_dips) or (i=nof_peaks);
  for i:=0 to l1-1 do emax[i]:=lmax[i];
  nof_peaks:=l1;

  write('Max of max between two min');
  readln;
  write('Max : ');writeln;
  for i:= 0 to nof_peaks-1 do write(emax[i]:4);
  writeln;writeln;
```

```
  for i:=0 to nof dips-1 do write(emin[i]:4);
  writeln;

(* Rejects max less than 20% of maxx *)
  i:=0;
  k:=0;
  repeat

    if i<nof_peaks then
      begin
        lmax[k]:=emax[i];
        i:=i+1;
      (*  while (i<nof peaks) and (emax[i]-lmax[k]<10) do
        begin

          i:=i+1;
        end; *)
        k:=k+1;
      end;
  until i=nof peaks;
  for i:=0 to k-1 do emax[i]:=lmax[i];
  nof_peaks:=k;

  write('20%');
  readln;
  write('Max : ');writeln;
  for i:= 0 to nof peaks-1 do write(emax[i]:4);
  writeln;writeln;
  for i:=0 to nof_dips-1 do write(emin[i]:4);
  writeln;

  i:=0;
  i1:=1;
  repeat
    while (i1<nof_dips) and (emin[i1]<emax[i]) do i1:=i1+1;
    i:=i+1;
    emin[i]:=emin[i1];
    while (i1<nof_dips) and (emin[i1]<emax[i]) do
      begin

        i1:=i1+1;
      end;
  until i=nof peaks;

  emin[nof_peaks+1]:=emin[nof_dips-1];

  nof_dips:=nof peaks+1;
```

```
  write('Max : ');writeln;
  for j:= 0 to nof_peaks-1 do write(emax[i]:4);
  writeln;writeln;
  for i:=0 to nof_dips-1 do write(emin[i]:4);
  writeln;


  i:=0;
  i1:=0;
  j1:=0;
  repeat
    spread[i]:=emin[i+1]-emin[i];
    i:=i+1;
  until i=nof_peaks;

  write('Max : ');
  for i:= 0 to nof_peaks-1 do write(emax[i]:4);
  writeln;
  write('Min : ');
  for i:=0 to nof_dips-1 do write(emin[i]:4);
  writeln;
  write('Dur : ');
  for i:=0 to k-1 do write(spread[i]:4);
  writeln;readln;

end;
```

Appendix E

Arabic Vowel Recognition and Similarity Measures Routines

```
(*koutines for program vo_recl.pas*)

procedure euc(n:integer);
label 10,20,30,40,50,60,70;
const
g:integer=6;
{n:integer=3;}
var
f1:text;
i,j,l,k,dd,h,r,n1,g1,g2,g3,g4,g5,g6          :integer;
data,tt,pp,e,app             :real;
a:array [1..50,1..6]of real;
x:array [1..50] of real;
dis:array [1..6] of real;
m:array [1..6] of integer;
  begin
      assign(fi,name);
      reset(fi);
      {calculate mean }
      {clear arrays}
      tt:=0;
      for i:=1 to g do
        begin
          for j:=1 to n do a[j,i]:=0;
        end;
      for I:=1 to g do m[i]:=27;{group}
      for k:=1 to g do
        begin
          dd:=m[k];
          for j:=1 to dd do
            begin
              for i:=1 to n do
                begin
                  read(fi,data);
                  a[i,k]:=a[i,k]+data;
                end;
            end;
        end;

      {get mean}
      for k:=1 to g do
        begin
          for i:=1 to n do
          begin
           a[i,k]:=a[i,k]/m[k];
(**        write(a[i,k]:7:4,    ');     **)
          end;
(**       writeln;                      **)
```

```
end;
 close(fi);
 assign(fi,name1);
 {*** end print***}
 {*** classify data***}
 {graphmode;textmode;}
 if name='harfor1' then
                        begin
                         gotoxy(2,3);

                        end;
 if name='harfft1' then
                        begin
                         gotoxy(2,3);

                        end;
 if name='harauto1' then
                        begin
                         gotoxy(2,3);

                        end;
 if name='harlpc1' then
                        begin
                         gotoxy(2,3);

                        end;
 if name='harref1' then
                        begin
                         gotoxy(2,3);

                        end;
 if name='harceps1' then
                        begin
                         gotoxy(2,3);

                        end;

 gotoxy(2,1);write('ARABIC VOWEL RECOGNITION');

{ gotoxy(2,6);write('DIST_MEASURE');
 gotoxy(3,10);write('EUCLIDEAN');}
 gotoxy(35,6);write('RECOGNIZED AS');
 gotoxy(34,4);write('CONFUSION  MATRIX');
 gotoxy(10,10);write('FATHAH');
 gotoxy(5,11);write('LONG_FATHAH');
 gotoxy(10,12);write('DAMMAH');
 gotoxy(5,13);write('LONG_DAMMAH');
 gotoxy(10,14);write('KASRAH');
```

```
gotoxy(5,5);write('LONG KASRAH');
gotoxy(17,6);write('FATHAH');
gotoxy(25,7);write('LONG');
gotoxy(25,8);write('FATHAH');
gotoxy(33,8);write('DAMMAH');
gotoxy(40,7);write('LONG');
gotoxy(40,8);write('DAMMAH');
gotoxy(48,8);write('KASRAH');
gotoxy(56,7);write('LONG');
gotoxy(56,8);write('KASRAH');
gotoxy(66,8);write('% RECOGNITION');
gotoxy(2,9);write('S');
gotoxy(2,10);write('P');
gotoxy(2,11);write('O');
gotoxy(2,12);write('K');
gotoxy(2,13);write('E');
gotoxy(2,14);write('N');
gotoxy(2,15);write(' ');
gotoxy(2,16);write('A');
gotoxy(2,17);write('S');

reset(fi);e:=0;n1:=0;writeln;tt:=27*g;
n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
for h:=1 to g do
  begin
    dd:=m[h];
    for r:=1 to dd do
     begin
      for i:=1 to n do read(fi,x[i]);
      for i:= 1 to g do dis[i]:=0;
      for k:=1 to g do
       begin
          for i:=1 to n do
            begin

            end;
       end;

       app:=dis[1];j:=1;
       for i:=2 to g do
         begin
          if dis[i]<=app then
                        begin
                          app:=dis[i];
                          j:=i;
                        end;
         end;
         if j=1 then g1:=g1+1;
```

```
                    if j=2 then g2:=g2+1;
                    if j=3 then g3:=g3+1;
                    if j=4 then g4:=g4+1;
                    if j=5 then g5:=g5+1;
                    if j=6 then g6:=g6+1;

                    if j<>h then
                            begin
                             e:=e+1;
                             n1:=n1+1;
                            end;
                end;

                gotoxy(70,9+h);
                write(100-100*n1/m[h]:4:2);
                n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
            end;
            gotoxy(49,17);
            write('TOTAL % RECOGNTION =  ',100-100* e/tt:4:2);

          close(fi);
      end;

procedure cor(n:integer);
 label 10,20,30,40,50,60,70;
 const
 g:integer=6;
 {n:integer=3;}
 var
 fi:text;
 i,j,l,k,dd,h,r,n1,g1,g2,g3,g4,g5,g6          :integer;
 data,tt,pp,e,app,aa,bb,cc          :real;
 a:array [1..50,1..6]of real;
 x:array [1..50] of real;
 dis:array [1..6] of real;
 m:array [1..6] of integer;

   begin
      assign(fi,name);
      reset(fi);
      {calculate mean }
      {clear arrays}
      tt:=0;
      for i:=1 to g do
        begin
          for j:=1 to n do a[j,i]:=0;
        end;
      for I:=1 to g do m[i]:=27;{group}
```

```
        for k:=1 to g do
          begin
            dd:=m[k];
            for j:=1 to dd do
              begin
                for i:=1 to n do
                  begin
                    read(fi,data);
                    a[i,k]:=a[i,k]+data;
                  end;
              end;
          end;

        {get mean}
        for k:=1 to g do
          begin
            for i:=1 to n do
            begin
              a[i,k]:=a[i,k]/m[k];
(**           write(a[i,k]:7:4,'  ');   **)
            end;
(**         writeln;                        **)
          end;
          {*** end print***}
          close(fi);
          assign(fi,name1);
           {*** classify data***}
           reset(fi);e:=0;n1:=0;writeln;tt:=27*g;
           n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
           for h:=1 to g do
             begin
               dd:=m[h];
               for r:=1 to dd do
                begin
                 for i:=1 to n do read(fi,x[i]);

                 for k:=1 to g do
                  begin
                    for i:=1 to n do
                      begin
                        aa:=aa+(x[i])*(a[i,k]);
                        bb:=bb+(x[i])*(x[i]);
                        cc:=cc+(a[i,k])*(a[i,k]);
                      end;
                      dis[k]:=aa/(sqrt(bb)*sqrt(cc));
                      aa:=0;bb:=0;cc:=0;
                  end;
```

```
                app:=dis[1];j:=1;
                for i:=2 to g do
                  begin
                    if dis[i]>app then
                                  begin
                                    app:=dis[i];
                                    j:=i;
                                  end;
                  end;
                    if j=1 then g1:=g1+1;
                    if j=2 then g2:=g2+1;
                    if j=3 then g3:=g3+1;
                    if j=4 then g4:=g4+1;
                    if j=5 then g5:=g5+1;
                    if j=6 then g6:=g6+1;

                    if j<>h then
                             begin
                              e:=e+1;
                              n1:=n1+1;
                             end;
              end;

              gotoxy(70,9+h);
              write(100-100*n1/m[h]:4:2);
              n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
            end;
            gotoxy(49,17);
            write('TOTAL % RECOGNTION = ',100-100* e/tt:4:2);

          close(fi);
      end;

procedure man(n:integer);
label 10,20,30,40,50,60,70;
const
g:integer=6;
var
fi:text;
i,j,l,k,dd,h,r,n1,g1,g2,g3,g4,g5,g6          :integer;
data,tt,pp,e,app,aa,bb,cc          :real;
a:array [1..50,1..6]of real;
s:array [1..50,1..50] of real;
x,tem,
t:array [1..50] of real;
dis:array [1..6] of real;
m:array [1..6] of integer;
```

```
begin
    assign(fi,name);
    reset(fi);
    {calculate mean }
    {calculate mean & covariance matrix}
    {clear arrays}
    tt:=0;
    for i:=1 to g do
      begin
        for j:=1 to n do a[j,i]:=0;
      end;
        for i:=1 to n do
          begin
            for j:=1 to n do s[i,j]:=0;
          end;
         for i:=1 to n do
          begin
            t[i]:=0;
            x[i]:=0;
          end;
      {end clear}
    for i:=1 to g do m[i]:=27;{group}
    for k:=1 to g do
      begin
        dd:=m[k];
        for j:=1 to dd do
          begin
            for i:=1 to n do
              begin
                read(fi,data);
                a[i,k]:=a[i,k]+data;
                x[i]:=data;

              end;
            for i:=1 to n do
              begin

              end;
          end;
      end;
    {correct x_product}
    for k:=1 to g do
      begin
        for j:=1 to n do
          begin

          end;
```

```
            tt:=tt+m[k];
          end;
        {finish covariance}
        for j:=1 to n do
          begin
            for i:=1 to j do
              begin
                s[i,j]:=s[i,j]/(tt-g);
                s[j,i]:=s[i,j];
              end;
          end;
        {get mean}
        for k:=1 to g do
          begin
            for i:=1 to n do
            begin
              a[i,k]:=a[i,k]/m[k];
(**             write(a[i,k]:7:4,'  ');  **)
            end;
           { writeln;}
          end;
            {writeln;writeln;writeln('covariance ');}
            for i:=1 to 2 do
             begin

              { writeln;}
             end;
           {****find inverse****}
               for i:=1 to n do
                 begin

                                                  else s[i,j]:=0;
                 end;
              {writeln;writeln;writeln('inverse');}
              for i:=1 to 2 do
               begin

                { writeln;}
           end;

            {*** end print***}
            close(fi);
            assign(fi,name1);
            {*** classify data***}
            n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
            reset(fi);e:=0;writeln;for i:=1 to n do tem[i]:=0;
            for h:=1 to g do
              begin
```

```
 dd:=m[h];
 for r:=1 to dd do
  begin
    for i:=1 to n do read(fi,x[i]);

    for k:=1 to g do
     begin
      for r:=1 to n do
       begin
        for i:=1 to n do
         begin

         end;
       end;


     end;

     app:=dis[1]; j:=1;
     for i:=2 to g do
       begin
        if dis[i]<=app then
                       begin
                         app:=dis[i];
                         j:=i;
                       end;
       end;
       if j=1 then g1:=g1+1;
       if j=2 then g2:=g2+1;
       if j=3 then g3:=g3+1;
       if j=4 then g4:=g4+1;
       if j=5 then g5:=g5+1;
       if j=6 then g6:=g6+1;

       if j<>h then
                 begin
                  e:=e+1;
                  n1:=n1+1;
                 end;
    end;

    gotoxy(70,9+h);
    write(100-100*n1/m[h]:4:2);
    n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
end;
gotoxy(49,17);
write('TOTAL % RECOGNTION = ',100-100* e/tt:4:2);
```

```
        close(fi);
    end;

procedure man1(n:integer);
  label 10,20,30,40,50,60,70;
  const
  g:integer=6;
  {n:integer=3;}
  var
  fi:text;
  i,j,l,k,dd,h,r,n1,q1,q2,q3,q4,q5,q6          :integer;
  data,tt,pp,e,app,aa,bb,cc          :real;
  a:array [1..25,1..6]of real;
  s:array [1..25,1..25] of real;
  x,tem,
  t:array [1..25] of real;
  f,d,dis,
  p:array [1..6] of real;
  m:array [1..6] of integer;
  v:array [1..25,1..150] of real;

    begin
      assign(fi,name);
      reset(fi);
      {calculate mean & covariance matrix}
      {clear arrays}
      tt:=0;
      for i:=1 to g do
        begin
          for j:=1 to n do a[j,i]:=0;
        end;
          for i:=1 to n do
            begin
              for j:=1 to n do s[i,j]:=0;
            end;
           for i:=1 to n do
             begin
               t[i]:=0;
               x[i]:=0;
             end;
           for i:=1 to n*g do
             begin
               for j:=1 to n do v[j,i]:=0;
             end;
        {end clear}
      for i:=1 to g do m[i]:=27;{group}
      for i:=1 to g do p[i]:=1/6;
       {calculate mean & covariance matrix}
```

```
for k:=1 to g do
  begin
    dd:=m[k];
    for j:=1 to dd do
      begin
        for i:=1 to n do
          begin
            read(fi,data);
            a[i,k]:=a[i,k]+data;
            x[i]:=data;

          end;
        for i:=1 to n do
          begin
            for l:=1 to i do

          end;
      end;
  end;
{correct x product}
for k:=1 to g do
  begin
    for j:=1 to n do
      begin
        for i:=1 to j do
          begin

            v[j,i+n*(k-1)]:=v[i,j+n*(k-1)];
          end;
      end;
      tt:=tt+m[k];
  end;
{get mean}
for k:=1 to g do
  begin
    for i:=1 to n do
    begin
    a[i,k]:=a[i,k]/m[k];
(**       write(a[i,k]:7:4,'     '); **)
    end;
    {writeln;}
  end;
 { writeln;writeln;}
  {***print covariance matrices}
    for k:=1 to g do
     begin
       for i:=1 to n do
```

```
      begin

        {writeln;}
       end;
       {writeln;}
    end;
{****end print*****}
{*** find inverse***}
  for k:=1 to g do
   begin
       for i:=1 to n do
         begin
            for j:=1 to n do
               begin
                  if i<>j then v[i,j+n*(k-1)]:=0

                end;
          end;
     end;
{***print inverse***}
    for k:=1 to g do
      begin
         for i:=1 to n do
          begin

           {writeln;}
          end;
          {writeln;}
          {writeln;}
       end;
{** end of print ***}

close(fi);
assign(fi,name1);
{*** classify data***}
n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
reset(fi);e:=0;writeln;for i:=1 to n do tem[i]:=0;
for h:=1 to g do
  begin
    dd:=m[h];
    for r:=1 to dd do
     begin
        for i:=1 to n do read(fi,x[i]);

       for k:=1 to g do
        begin
           for i:=1 to n do
             begin
```

```
                end;

            end;

            app:=dis[1];j:=1;
            for i:=2 to g do
              begin
                if dis[i]<=app then
                              begin
                                app:=dis[i];
                                j:=i;
                              end;
              end;
              if j=1 then g1:=g1+1;
              if j=2 then g2:=g2+1;
              if j=3 then g3:=g3+1;
              if j=4 then g4:=g4+1;
              if j=5 then g5:=g5+1;
              if j=6 then g6:=g6+1;

              if j<>h then
                       begin
                         e:=e+1;
                         n1:=n1+1;
                       end;
            end;

            gotoxy(70,9+h);
            write(100-100*n1/m[h]:4:2);
            n1:=0;g1:=0;g2:=0;g3:=0;g4:=0;g5:=0;g6:=0;
        end;
        gotoxy(49,17);
        write('TOTAL % RECOGNTION = ',100-100* e/tt:4:2);

    close(fi);
end;
```

Appendix F

Dynamic Time Warping Routines

```
          (* Trainig mode*)
          procedure create_templates;
          label 11,12;
          var
           i,j,k                :integer;
           fy,fk                   :string[20];
           fi,fd:text;
           begin
              textmode;clrscr;
              if pointer =1 then goto 11;
              fy:='ztest.dat';
              fk:='etest.dat';
              goto 12;
          11: writeln('Input Speech Templates Names ');
              readln(fy);
              readln(fk);
          12: assign(fi,fy);
              assign(fd,fk);
              rewrite(fd);
              rewrite(fi);
              i:=adur-1;
              writeln(fi,j);
              writeln(fd,j);
               for i:=0 to j do
                         begin
                              writeln(fi,round(zcr[i]/zmax*100));
                              writeln(fd,round(sum[i]/maxx*100));
                         end;
              close(fi);
              close(fd);
              clrscr;
           end;
(* Dynamic time Warping technique*)
procedure match2;

const
  size = 110;
  maxy = 10;

type
  arr_size = array [0..size] of integer;

var
  restr        : integer;
  name         :string[10];
  fi           : text;
  x_array      : arr_size;
  y_array      : array [0..maxy] of arr_size;
```

```
  cost          : array [1..3] of integer;
  q,
  d             : array [0..size] of arr_size;
  i,j,k,p,
  not_plates,
  test_len,
  nofx,
  nofy          : integer;
  data          : text;
  rer           :array[0..10]of real;
  smallest      :real;

procedure get_data;

var
   i,
   j,
   len : integer;

begin
   for i:=1 to size do
     begin
       d[i,0]:=32000;
       q[i,0]:=32000;
     end;
   for i:=1 to size do
     begin
       d[0,i]:=32000;
       q[0,i]:=32000;
     end;

   writeln(' X-Data - Plate6');
   writeln('Enter test Syllabe');
   readln(name);
   assign(fi,name);
   reset(fi);
   readln(fi,test_len);
   for i:= 1 to test_len do
     begin
       readln(fi,x_array[i]);
     end;
   close(fi);
     (******)
   (*
   writeln(' Y-Data - Plate0');
   *)
   assign(fi,'td0.e ');
   reset(fi);
```

```
  read(fi,len);
  y_array[0,0]:=len;
  for j:=1 to len do read(fi,y_array[0,j]);
  close(fi);
  (*
  writeln(' Y-Data - Plate1');
  *)
  assign(fi,'td1.e ');
  reset(fi);
  readln(fi,len);
  y_array[1,0]:=len;
  for j:=1 to len do readln(fi,y_array[1,j]);
  close(fi);

(******)
  (*
  writeln(' Y-Data - Plate2');
  *)
  assign(fi, 'td2.e ');
  reset(fi);
  read(fi,len);
  y_array[2,0]:=len;
  for j:=1 to len do read(fi,y_array[2,j]);
  close(fi);
(*******)
  (******)
  (*
  writeln(' Y-Data - Plate3');
  *)
  assign(fi,'td3.e ');
  reset(fi);
  read(fi,len);
  y_array[3,0]:=len;
  for j:=1 to len do read(fi,y_array[3,j]);
  close(fi);
(******)
  (******)
  (*
  writeln(' Y-Data - Plate4');
  *)
  assign(fi,'td4.e');
  reset(fi);
  read(fi,len);
  y_array[4,0]:=len;
  for j:=1 to len do read(fi,y_array[4,j]);
  close(fi);
  (******)
  (*
```

```
writeln(' Y-Data - Plate5 ');
*)
assign(fi,'td5.e');
reset(fi);
read(fi,len);
y_array[5,0]:=len;
for j:=1 to len do read(fi,y_array[5,j]);
close(fi);
  (*****)
(*
writeln(' Y-Data - Plate6 ');
*)
assign(fi,'td6.e');
reset(fi);
read(fi,len);
y_array[6,0]:=len;
for j:=1 to len do read(fi,y_array[6,j]);
close(fi);
  (*****)
(*
writeln(' Y-Data - Plate7');
*)
assign(fi,'td7.e');
reset(fi);
read(fi,len);
y_array[7,0]:=len;
for j:=1 to len do read(fi,y_array[7,j]);
close(fi);
  (*****)
(*
writeln(' Y-Data - Plate8');
*)
assign(fi,'td8.e');
reset(fi);
read(fi,len);
y_array[8,0]:=len;
for j:=1 to len do read(fi,y_array[8,j]);
close(fi);
  (*****)
(*
writeln(' Y-Data - Plate9');
*)
assign(fi,'td9.e');
reset(fi);
read(fi,len);
y_array[9,0]:=len;
for j:=1 to len do read(fi,y_array[9,j]);
close(fi);
```

```
    (*****)
  (*
  writeln(' y-Data - Plate10');
  *)
  assign(fi, 'd10.e');
  reset(fi);
  read(fi,len);
  y_array[10,0]:=len;
  for j:=1 to len do read(fi,y_array[10,j]);
  close(fi);

  nof_plates:=11;
end;

procedure compute_dist(x,y:integer);

var
  i     : integer;

begin
  if x=2 then cost[1]:=g[x,y-1] + d[x,y]


  if y=2 then cost[2]:=g[x-1,y] + d[x,y]


  cost[3]:=g[x-1,y-1] + 2*d[x,y];   (* diagonal *)

(*  write('Path Dist : ');
  for i:=1 to 3 do write(cost[i],' ');
  writeln;
*)end;


function find_min:integer;

var
  i,
  min : integer;

begin
  min:=1;
  for i:=1 to 3 do
    if cost[min]>cost[i] then min:=i;
  if cost[min]<0 then writeln('Min ',min);
(*  writeln('Min : ',min);
*)  find_min:=cost[min];
end;
```

```
procedure generate_d(index,len:integer);

var
   i,
   j       : integer;

begin
   (*write(index,' ',len,' ',test_len,'   ');*)
   for i:=1 to len do
     for j:=1 to test_len do
      begin

        if d[j,i] < 0 then writeln('Neg d Matrix');
      end;
{**  writeln('d Matrix');
   for i:=len downto 1 do
     begin

            x_array[j]:10:2);
       writeln;
     end;**)
end;


procedure generate_g(len:integer);

var
   i,
   j       : integer;

begin
   g[1,1]:=d[1,1];
   for i:=2 to len do g[1,i]:=g[1,i-1]+d[1,i];
   for i:=2 to test_len do g[i,1]:=g[i-1,1]+d[i,1];



   for i:=2 to len do
     for j:=2 to test_len do
         begin
           compute_dist(j,i);
           g[j,i]:=find_min;
           if g[j,i] < 0 then
             begin
               writeln('Neg g Matrix');
               writeln(j:4,i:4,'   ',d[j,i],'   ',g[j,i],'   ',
```

```
                        g[j-1,i-2],'  ',g[j,i-1]);
            end;
        end;
 {** writeln('g Matrix');
  for i:=len downto 1 do
    begin
      for j:=1 to test_len do write(g[j,i]:10:2);
      writeln;
    end;**)
end;


procedure find_path(index:integer);

var
  i,xx,yy,xxx,yyy,
  j,
  k,
  len  : integer;
  prev_cost,
  path_cost : real;
  path : array [1..300] of integer;

begin
  len:=round(y_array[index,0]);
  restr:=1;
  path_cost:=0.0;
  prev_cost:=1.0;
while (restr<>len) and (prev_cost<>path_cost) do
begin
  if restr<>1 then prev_cost:=path_cost;
  i:=1;
  j:=1;
  k:=1;
  path_cost:=g[1,1];
  while (i<>test_len) and (j<>len) do
    if abs(i-j) <= restr then
    begin


      if (path[k-2]<>1) and (path[k-1]<>1) then path[k]:=1;
      {**else if (i<j) and (path[k]=2) then path[k]:=3;**)
      if path[k]=2 then j:=j+1
      else if path[k]=3 then i:=i+1
      else
        begin
          i:=i+1;
```

```
            j:=j+1;
          end;
        path_cost:=path_cost+g[i,j];
        k:=k+1;
      end
      else
        if i>i then
          begin
            j:=j+1;
            path[k]:=2;
            k:=k+1;
            path_cost:=path_cost+g[i,j];
            end
          else
          begin
            i:=i+1;
            path[k]:=3;
            k:=k+1;
            path_cost:=path_cost+g[i,j];
          end;
    if i<>test_len then
      while i<test_len do
        begin
          i:=i+1;
          path_cost:=path_cost+g[i,j];
          path[k]:=3;
          k:=k+1;
        end
    else
      while j<len do
        begin
          j:=j+1;
          path_cost:=path_cost+g[i,j];
          path[k]:=2;
          k:=k+1;
        end;
    restr:=restr+1;
  end;
   (* write('Path : ');*)
    graphmode;
    xxx:=20;yyy:=190;xx:=20;yy:=190;
    for i:= 1 to k-1 do
              begin
                if path[i]=1 then
                                  begin
                                    xx:=xx+3;
                                    yy:=yy-1;
                                  end;
```

```
                if path[i]=3 then xx:=xx+3
                             else yy:=yy-1;
                draw(xxx,yyy,xx,yy,1);
                xxx:=xx;
                yyy:=yy;
              end;
              draw(20,190,20,yy,3);
              draw(20,190,xx,190,3);
              draw(20,yy,xx,yy,3);
              draw(xx,190,xx,yy,3);
              xxx:=20;
              yyy:=190;
              repeat
                  yyy:=190;
                  repeat
                    plot(xxx,yyy,2);
                    yyy:=yyy-2;
                  until yyy<=yy;
                  xxx:=xxx+3;
              until xxx=xx;
  xxx:=20;yyy:=190;xx:=20;yy:=190;
  for i:= 1 to k-1 do
            begin
              if path[i]=1 then
                             begin
                               xx:=xx+3;
                               yy:=yy-1;
                             end;
                if path[i]=3 then xx:=xx+3
                             else yy:=yy-1;
                draw(xxx,yyy,xx,yy,1);
                xxx:=xx;
                yyy:=yy;
              end;

  readln;
  (*rec[index]:=path_cost/test_len;*)

end;


begin
  get_data;
  for i:= 0 to nof_plates-1 do
    begin
      generate_d(1,round(y_array[i,0]));
      generate_g(round(y_array[i,0]));
      find_path(i);
```

```
        end;
    end;
```

Appendix G

Broad Phonetic Decoding Routines

```
(*broad phonetic decoding*)
procedure lattice;
  var
  i,j,ii,nof max :integer;
  form:array[1..4]or integer;


  procedure lpc coef(a max:integer);
const
n:integer=256;
p:integer=14;
h:real=2.302851;
var
  i,frame,
  l          : integer;
  be,ww,pt,v        :real;
  wdata,spec:array[0..256]of real;
  r,e,k1,row:array[0..15]of real;
  alfa:array[0..15,0..15]of real;
begin
  {programe to calculate lpc coef 15}
  {writeln( input frame number: ');
  readln(frame);(*multiplay by window*);}
  frame:=(a max)*64+sframe;
  {writeln(frame);}


  {now calculate autocorraltions coefficents in R[0..p]}

    for i:=0 to p do
    begin
     R[i]:=0;
      for j:=0 to N-1-i do R[i]:=R[i]+wdata[j]*wdata[j+i]
    end;
  {recursive procedure Durbins for lp}
    be:=0;
    E[0]:=R[0];
    for i:=1 to 14 do
    begin
       for j:=1 to i-1 do
      begin
       be:=be+alfa[(i-1),j]*R[i-j];
      end;
      K[i]:=-(R[i]+be)/E[i-1];
      alfa[i,i]:=K[i];
       for j:=1 to i-1 do
        begin
         alfa[i,j]:=alfa[i-1,j]+K[i]*alfa[(i-1),(i-j)];
```

```
      end;
    E[i]:=(1-sqr(k[i]))*E[i-1];
    be:=0;
  end;
  {cal Gain V}
  V:=0;
  for j:=1 to 14 do V:=V+alfa[14,i]*R[i];
  V:=V+R[0];
 {for i:=1 to 14 do writeln(alfa[14,i]);}
 {now plot spectrum}
  for i:=0 to 14 do row[i]:=0;
  alfa[14,0]:=1;
  for i:=0 to 14 do
   begin
    for j:=0 to 14-i do
     begin
      row[i]:=row[i]+alfa[14,j]*alfa[14,j+i];
     end;
   end;
 {spectrum}
  ww:=0;pi:=0;
  i:=0;
  repeat
   for i:=1 to 14 do
     begin
      ww:=ww+row[i]*cos(i*pi*3.14159/5000);
     end;
   pi:=pi+20;
   spec[j]:=ln(sqr(V)/(row[0]+2*ww));
   j:=j+1;
   ww:=0;
  until j>249;
  {graphmode;
  for i:=0 to 249 do plot(i,400-round(10*spec[i]),1);}
  {for i:=0 to 99 do writeln(round(10*spec[i]));}
for i:=0 to 8 do
  begin
    lmax[i]:=-1;
    lmin[i]:=-1;
  end;
i:=0;
j:=0;
repeat
  while (i<249) and (lmax[j]<0) do
    begin
      if (spec[i]) <= (spec[i+1]) then i:=i+1
      else lmax[j]:=i;
    end;
```

```
    if (i=249) and (lmax[j]=-1) then lmax[j] := 249;
    i := i + 1;
    while (i<249) and (lmin[j]<0) do
      begin
        if (spec[i]) >= (spec[i+1]) then i:=i+1
        else lmin[j]:=i;
      end;
    if (i=249) and (lmax[j]<>-1) then lmin[j] := 249;
    i := i + 1;
    j := j+1;
  until i>=250;
  nof_max:=j-1;
end;

 begin
  textmode;
  gotoxy(15,10);
  write( Recognized No Of Syllabels=', nof_peaks);
  if nof_peaks>=3 then
                    begin
                      j:=0;
                      for i:=1 to nof_peaks-1 do
                          begin
                            if sum[i]>sum[j] then j:=i;
                          end;
                    end
                  else
                    begin
                      j:=0;
                      for i:=1 to nof_peaks-1 do
                          begin

                          end;
                    end;
  gotoxy(15,11);
  write('Stressed Syllable is number ',j+1);
  for i:=1 to nof_peaks-1 do
                           begin
                             gotoxy(15,11+i);



                           end;
 gotoxy(15,15);
 for i:=0 to nof_dips-1 do
                     begin

                     end;
```

```
gotoxy(15,16);
peak_detection;
 for i:=0 to nof_peaks-1 do
  begin
    gotoxy(15,16+1);
    write( Vowel ',i+1,' formants are ');
    lpc_coef(emax[i]);

    j:=0;
    for i1:=0 to nof_max do
                begin
                  if (lmax[i1]*20>=200) then
                                        begin
                                         j:=j+1;

                                        end;
                end;
      if form[1]>700 then write('Rejected')
                     else
                      for i1:=1 to 4 do write(form[i1],'   ')


    end;

  gotoxy(15,21);
  if nof_peaks=1 then
                   begin

                                       else write('M');
                   end
                 else
                  begin
                   for i:=0 to nof_peaks-1 do
                    begin


                     end;
                  end;
end;
```

REFERENCES

1. Anees, I., **"Arabic Sounds"**, Anglo-Egyptian Library, Cairo, Egypt, 1979.

2. Al-Antaki, M., **"Al-Mohit in Arabic Sounds and Morphology"**, Dar Alsarq, Lebanon, 1975.

3. Moussa, A. H., **"Occurrence Probability of Characters and Entropy of Arabic Language"**, Arab School of Science and Technology on "Informatics and Applied Arabic Linguistics, 7th Summer Session, pp. 30-34, Zabadani Valley, Syria, July 23-31, 1985.

4. Omar, A., **"Study of Linguistics"**, Alam AlKutub, Cairo, Egypt, 1981.

5. Ladefoged, P., **"A Course in Phonetics"**, Harcourt Brace Jovanovich, Inc., United Kingdom, 1982.

6. Biser, K., **"Arabic Sounds"**, Dar ElMauref, Cairo, Egypt, 1980.

7. Fredrick Williams., **"Language and Speech Introductory Perspective"**, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1972.

8. Flanagan, J. L., Coker, C. H., Rabiner, L. R., and Schafer, R. W., "Synthetic Voices for Computer", Bell Telephone Laboratories Inc., IEEE Spectrum, October 1970.

9. Descout, R., Arab School on Science and Technology on "Applied Arabic Linguistics and Signal, and Information Processing", 1st Fall Session, Rabat, Morocco, September 26 - October 5, 1983.

10. Jayant, N. S., and Noll, P., "Digital Coding of Waveforms, Principles and Applications to Speech and Video", Prentice Hall, New Jersey, 1980.

11. Linggard, R., "Electronic Synthesis of Speech", Cambridge University Press, Cambridge, 1985.

12. Makhoul, J., "Linear Prediction: A Tutorial Review", IEEE Proceeding, Vol. 63, pp. 561-580, April 1975.

13. Bristow, G., "Electronic Speech Synthesis: Techniques, Technology and Applications", Granada Technical Books, Granada Pub. Ltd., London, 1984.

14. Reddy, D. R., "Tutorial on System Organization for Speech Understanding", IEEE Symposium on Speech Recognition, 1974.

15. Klatt, D. H., "Structure of a Phonological Rule Component for a Synthesis by Rule Program", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-24 No. 5, October 1976.

16. Liberman, A. M., "Minimal Rules for Synthesizing Speech", The Journal of Acoustical Society of America, Vol. 31, Number 11, November 1959.

17. Elovitz, H. S., "Letters to Sound Rules for Automatic Translation of English Text to Phonetics", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-24 No. 6, December 1976.

18. Hertz, S. R., "SRS Text to Phoneme Rules: A Three Level Rule Strategy", ICASSP 81. Proceedings of the 1981 IEEE International Conference on Acoustics, Speech and Signal Processing, Atlanta, GA, USA, (New York, USA: IEEE 1981), pp. 102-105, Vol. 1, 30 March - 1 April, 1981.

19. Holmes, J. N., Mattingly, I. G., and Sheame, J. N., "Speech Synthesis by Rule", Language and Speech 7, pp. 127-43, 1964.

20. Olive, J. P., **"Rule Synthesis of Speech for Dyadic Units"**, IEEE International Conference on Acoustic, Speech, and Signal Processing, Rec., Hartford, Conn. May 9-11, 1977. New York IEEE (Cat. No. 77 CH1197-3 ASSP).

21. Mouradi, A., **"United Vocabulary Synthesis System for Arabic Language"**, LEESA, Faculte Des Sciences, B. P. 1014, Rabat, Morocco, 1983.

22. Dettweiler, H. and Hess, W., **"Concatenation Rules for Demisyllable Speech Synthesis"**, Acoustica, Vol. 57, pp. 268-283, 1985.

23. Moore, R. K., **"Systems for Isolated and Connected Word Recognition"**, NATO ASI Series, Vol. F16. New Systems and Architectures for Automatic Speech Recognition and Synthesis. Edited by R. DeMori Springer-Verlas Berlin Heidelberge, 1985.

24. Rabiner, L. R. and Sambur, M. R., **"An Algorithm for Determining the End Points of Isolated Utterances"**, Bell Sys. Tech. J., pp. 297-315, February 1975.

25. Baker, J. K., **"Stochastic Modelling for Automatic Speech Understanding"**, D. R. Reddy (ed.), Speech Recognition, pp. 521-542, Academic Press, 1975.

26. Moore, R. K., **"Overview of Speech Input"**. IFS International Conference on 'Speech Technology' Brighton, U.K., October 23-25, 1984.

27. Cole, R. A., et. al., **"Speech as Patterns on Paper"**, R. A. Cole (ed.), Perception and Production of Fluent Speech, Erlbaum, Hillsadle, 1980.

28. Mrayati, M., **"Electronic Speech Synthesis"**, Arab School on Science and Technology, 1st Fall Session, Rabat, Morocco, September 26 - October 5, 1983.

29. Mandurah, M. M., **"Synthesis of Arabic Speech Using Phoneme-Based Synthesizers"**, J. Eng. Science, King Saud University. Vol. 10 (1-2), 9-14, 1984.

30. Hashish, M. A., **"Experiences in Isolated Arabic Characters Recognition"**, Cairo Scientific Center, IBM, Egypt, Progress Report No. 8, August 1985.

31. Abdulla, W. H., **"Real Time Arabic Digit Recognizer"**, Intl. J. Electronics. Vol. 59, No. 5, pp. 645-648, 1975.

32. Mahmoud, W. A., "Recognition of Arabic Digits by a Microcomputer", Jordan International Electrical & Electronic Engineering Conference, April 28 - May 1, 1985. Amman, Jordan.

33. El-Imam, Y. A., "Speech Synthesis by Concatenation of Sub-syllabic Sound Units", IEEE-ICASP '87, Dallas, Texas, Vol. 4, 1987.

34. Data Translation, Inc., 100 Locke Drive, Marlborough, Massachusetts, DT2801. Data Acquisition Board User Manual.

35. Rabiner, L. R., and Schafer, R. W., "Digital Processing of Speech Signals", Prentice Hall International, 1978.

36. Wolf, J. J., "Speech Signal Processing and Feature Extraction", J. C. Simon (ed.), Spoken Language Generation and Understanding, pp. 103-128, D. Reidel Publishing Company, 1980.

37. Oppenheim, A. V. and Schafer, W. R., "Digital Signal Processing", Prentice Hall Inc., New Jersey, 1975.

38. Signal Technology Inc., "Interactive Laboratory System" 5951 Encina Road, Goleta, California 93117, USA, 1985.

39. Texas Instrument Inc., Data System Group "TI-Speech" Signal Processing Board. P. O. Box 2909 M/S 2234, Austin, TX 78769-2909, USA, 1985.

40. Delattre, P. C., Liberman, A. M., and Cooper, F. S. Acoustic loci and transitional cues for consonants. J. Acoust. Soc. America, 27, 769, 1955.

41. Fry, D. B. Duration and intensity of physical correlates of linguistic stress. J. Acoust. Soc. America, Vol. 27, No. 4, 1954.

42. Tecmar, Inc., PC Product Division, 6225 Cochran Road, Solon (Cleveland), Ohio 44139, USA. PC-Graphic Master Users Guide, 1985.

43. Digital Signal Processing Software, Inc., "Realtime Signal Laboratory", P. O. Box 5348, Station F, Ottawa, Canada K2C 3J1, 1985.

44. Reddy, D. R. Segmentation of Speech Sounds, Journal of the Acoustical Society of America, 1966.

45. Benedetto, M. D. and Destombes, F., "Phonetic Recognition to Assist Lip-reading for Deaf Children" Proceedings ICASSP '82, IEEE International Conference on Acoustic, Speech and Signal Processing.

46. Mermelstein, P., "Automatic Segmentation of Speech into Syllabic Units", J. Acoust. Soc. Am., Vol. 58, No. 4, pp. 880-883, October 1975.

47. Bertil Malmberg, "Phonetics", Dover Publications, Inc., 1983.

48. Carter, J. P., "Electronically Hearing: Computer Speech Recognition", Howard W. Sam & Co. Inc., Indiana, USA, 1984.

49. Chi-Tsong Chen, "One-Dimensional Digital Signal Processing", Marcel Dekker, Inc., New York and Basel, 1979.

50. Atal, B. S., and Hanaver, S. L., "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", J. Acoust. Soc. Am., Vol. 50. No 2, pp. 637-655, 1971.

51. Fukunaga, K., "Introduction to Statistical Pattern Recognition", pp. 260-300, Academic Press, New York, 1972.

52. Duda, R., "Pattern Classification and Science Analysis", pp. 116-121, John Wiley & Sons Publication, 1973.

53. SAS Institute Inc., "SAS USER'S GUIDE: Statistics", 1982 Edition, Box 800 Cary, North Carolina 27511.

54. Batchelor B. G., "Pattern Recognition", Plenum Press, New York, 1978.

55. James, M., "Classification Algorithms", Collins Professional and Technical Books, 8 Grafton Street, London W1, 1985.

56. Parsons, T., "Voice and Speech Processing", McGraw-Hill Book Company, 1987.

57. Rabiner, L. R. and Levinson, S. E. Isolated and Connected Word Recognition-Theory and Selected Applications. IEEE Trans. Commun., 29, 621-659, 1981.

58. Shipman, D. W. and Zue, V, W. Properties of Large Lexicons: Implementations for Advanced Isolated Word Recognition Systems. Proc. IEEE International Conference on Speech Acoustics and Signal Processing, Paris, France, pp. 546-549, 1982.

59. Cutler, A. and Foss, D. J., "On the Role of Sentence Stress in Sentence Processing", Language and Speech, 20, pp. 1-10, 1977.

60. Levinson, N. The Wiener RMS Error in Filter Design and Prediction. J. Math. Phys. 25, 261-278, 91, 1974.

61. Rabiner, L. R. On Creating Reference Templates for Speaker Independent Recognition of Isolated Words. IEEE Trans. on Acoustic, Speech and Signal Processing, Vol. ASSP-26 No. 1, pp. 34-42, February 1978.

62. Davis S. B. and Mermelstein, P. Comparison of Parametric Representation for Monosyllabic Word Recognition in Continuously Spoken Sentences. IEEE Trans. on Acoustic Speech, and Signal Processing, Vol. ASSP-28 No. 4, pp. 357-366, August 1980.

63. Witten, I. H., **"Principles of Computer Speech"**, Academic Press, 1982.

64. Itakura, F. Minimum Prediction Residual Applied to Speech Recognition. IEEE Trans. on Acoustic, Speech and Signal Processing, Vol. ASSP-23 No. 1, pp. 67-72, February 1975.

65. H. Sakoe and Chida, S. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Trans. on Acoustic, Speech, and Signal Processing, Vol. ASSP-26, pp. 43-49, February 1978.

66. Christiansen R. W., **Detecting and Locating Keywords in Continuous Speech Using Linear Predicative Coding"**, IEEE Trans. on Acoustic., Speech and Signal Process., Vol. ASSSP-25, pp. 361-367, Oct. 1977.

67. Woods, W. A., **"Syntax, Semantics and Speech"**, D. R. Reddy (ed.), Speech Recognition, pp. 345-400, Academic Press, 1975.

68. Reddy, D. R., **"Tutorial on System Organization for Speech Understanding"**, D. R. Reddy (ed.), Speech Recognition, pp. 457-480, Academic Press, 1975.

69. Hans-Werner Hein, **"The Erlangen Speech Understanding Project"**, J. P. Haton (ed.), Automatic Speech Analysis and Recognition, pp. 239-251, D. Reidel Publishing Company, 1982.